Embodying conversational characteristics in a graphical user interface

Thesis

Embodying conversational characteristics in a graphical user interface

Ronald A. Singer, HND., ADCS.

Thesis submitted for the degree of
Doctor of Philosophy in Cognitive Science

The Open University, Milton Keynes

June, 1992

No man is an island, entire of itself,

every man is a piece of the continent,

a part of the main,

if a clod be washed away by the sea,

Europe is the less,

as well as if a premontory were,

as well as if a manor of thy friends

or of thine own were;

any mans death diminishes me,

because I am involved in Mankind;

And therefore never send to know

for whom the bell tolls;

it tolls for thee.

John Donne

1571 - 1631

Dedication

For my dear father and mother
to whom I owe everything

For my dear wife Deosmita Anjali

For Ken and Linda
who showed me the true meaning of
Nil Sine Laboure

EX12

HIGHER DEGREES OFFICE

LIBRARY AUTHORISATION FORM

STUDENT: RONALD A. SINGER. SERIAL NO: _____

DEGREE: Ph.D.

TITLE OF THESIS: EMBODYING CONVERSATIONAL CHARACTERISTICS IN A GRAPHICAL USER INTERFACE

I confirm that I am willing that my thesis be made available to readers and maybe photocopied, subject to the discretion of the Librarian.

SIGNED: DR. R.A. Singer DATE: 31-3-93.

Abstract

In the history of Intelligent Tutoring Systems, SOPHIE (Brown, Burton, and Bell, 1974), now considered a classic, contained many important ideas and features. One of these was its natural language user interface. Today, the trend has moved away from natural language interfaces towards graphical ones although the argument in favour of natural language user interfaces, both from Human Computer Interaction and natural language researchers, still persist. Is this argument correct?

This thesis explores this question by investigating how SOPHIE might be re-implemented with a graphical direct manipulation interface instead of a natural language one, with the goal of improving its standard of usability. It begins by analysing the features that seem to have been central to SOPHIE's usability. These, it argues, were not so much an ability to accept well formed complete English sentences, as an ability to accept and interpret correctly a wide range of abbreviated inputs.

Two models of interaction, Circuit I, a pilot, and Circuit II, a fairly full implementation of SOPHIE were implemented and tested. Both employ free-order syntax that allows users to specify the components of a full command in any order. The combination of deixis and free-order syntax supported allows completely general ellipsis which achieves, in extended interaction sequences, the same economy and naturalness that SOPHIE achieved through its use of anaphora and ellipsis.

Whilst the free-order syntax technique is little used at present in user interfaces, the results of observational studies conducted have shown that it saves users time and convenience. Thus, considering key linguistic features of a natural language user interface has shown how novel features can enhance the usability of direct manipulation interfaces. This

thesis argues that user interfaces can be improved by employing structures found in natural language or at least conversation which can be constructed within direct manipulation interface styles.

This approach was further expanded to support topic shifts between *different circuit contexts*. Circuit II, like SOPHIE, supports three different topics: normal circuit behaviour, a circuit with an unknown fault, and circuits with user-hypothesised faults. Drawing on Reichman's (1981) work, Circuit II uses natural language cue phrases of the type "by the way", re-implemented in the direct manipulation style, to facilitate shifts between topics in a smoother and more natural way than SOPHIE which used clumsy explicit commands.

Acknowledgements

Over the last four years at The Open University I have been fortunate to visit many places and to meet so many wonderful, considerate, and intelligent people. These individuals have been a great source of inspiration to me, who have guided my efforts to gain a doctoral degree. To these people I say thank you.

To Ann Jones, for her helpful advice and continuous encouragement throughout this work. I am very grateful for the countless hours spent by her patiently listening to my ideas.

To Tim O'Shea, whom I admire and respect, who first suggested SOPHIE as a model around which to build my work. I have learned so much from him, both as a researcher and human being.

To Chris Dillion, electronics expert, who spent countless hours discussing how he thought about the operation of electronic circuits. His advice and assistance was a valuable source of ideas for me.

To my external supervisor Steve Draper, for all the encouragement and technical advice he has given. His analysis of my work and timely comments have helped me avoid many pitfalls. His understanding of what it was I was trying to do, his guidance in the way it should be done all proved invaluable during my research. I am deeply grateful to him.

To Rachel Reichman, Stanford University, California, whose work I greatly admire. My discussions with her inspired me greatly to demonstrate that her work has an important role to play in making graphical interfaces easier to use.

To Alan Collins, Bolt Beranek Newman Inc., Cambridge, Massachusetts, who kindly responded to my initial enquiries concerning the resurrection of SOPHIE.

To Frank Zdybel, Xerox Palo Alto Research Center, California, for his advice concerning the resurrection of SOPHIE, and the difficulties involved in doing so. As one of the designer's of SOPHIE, he provided me with so much information about how I should proceed with my research. His input at such an early stage of my research was invaluable and I sincerely acknowledge his advice.

To fellow research colleagues, who read and commented on my thesis drafts: Paul Davidson, Maria Yannissi, Iraklis Parastakis, Magnus Moar, Mark Elsom-Cook, Claire O'Malley, Yibing Li, Matthew Stratfold, and Zhengmai Zhao.

To NATO Special Programme Panel on Advanced Educational Technology for their support of this work under contract number SA.9-14-04 (RV.900350) (90)LVdC. I am very grateful to Luis da Cuhna, Programme Director of the Special Programme on Advanced Educational Technology for his support during my contract.

I would like to thank members of CITE (past and present) for providing me with an exciting working environment and the opportunity to develop myself.

This work was supported by a studentship from the Open University.

Contents

FIGURES AND TABLES

Chapter Two

Chapter Three

Chapter Four

Chapter Six

Chapter Seven

Research Problem

1.1 The research problem

This thesis explores a way in which direct manipulation interfaces can be improved. The desire and need to communicate with computers on a conversational basis has long been established. In attempting to resolve this problem much attention has focussed on the area of natural language (for example, Hendrix, Sacerdoti, Sagalowicz, and Slocum, 1978; Bobrow, Kaplan, Kay, Norman, Thompson, and Winograd, 1986; Allen, 1987; Blandford, 1991). However, given the computational complexity of this approach many researchers have turned their attention to graphical interfaces which can provide the same functionality as a natural language one but without the complexity (for example, Robertson, McCracken, and Newell, 1981; Tennant, Ross, Saenz, Craig, and Miller, 1983; Loftin, Wang, Baffes, and Hua, 1989; Schute and Glaser, 1990). A number of graphical interfaces support the direct manipulation paradigm where the screen is divided into areas, icons can be moved around the screen and their behaviour controlled by pop-up or pull-down menus. A common complaint made by users of graphical interfaces is that within a given context the system does not always follow what they are doing from one mouse click to the next. Also, when moving to a different but related window context and then returning to the old window context, the system forgets what has been done previously which means that a user has to set up the context again (Reichman, 1986). The inability of graphical interfaces to do this seems to reduce the sense of coherency experienced by users during the interaction. This thesis investigates the feasibility of extending the direct manipulation paradigm by supporting conversational features within the interface. One of its aims is to see if by modelling such features users feel a greater sense of continuity and coherency during the interaction.

The central research problem addressed in this thesis is the reconstruction of the Intelligent Tutoring System, SOPHIE (Brown, Burton, and Bell, 1974), with a graphical rather than a natural language interface. One of SOPHIE's major contributions to the field of Intelligent Tutoring Systems was its use of semantic grammars developed by Burton (1975, 1976). This gave SOPHIE a number of distinct advantages over purely syntactic stratagems, such as allowing the system to accept a wider range of responses from the user, and increasing the ability of the parser to recognize irrelevant or missing words from a sentence.

Ellipsis

The use of semantic grammars as used in SOPHIE allows the system to recognize elliptic utterances, that is utterances that do not express complete thoughts, a complete question or command. Instead, only differences between the intended thought and an earlier one are given (Burton and Brown, 1979). For example, within the domain of electronics, statements 2, 3 and 4 below are elliptic utterances which SOPHIE handled.

1. What is the voltage at node 5?
2. At node 1?
3. And node 2?
4. What about between nodes 7 and 8?

and an example used in everyday conversation might be:

5. What is the time in the U.K.?
6. In Rome?
7. And in New York?
8. What about Moscow?

where statements 6, 7 and 8 are elliptic utterances.

Anaphoric reference

"In spoken and written discourse, people use certain words to 'point back' in the discourse to people, places, objects, times, events and ideas mentioned there. The use of such a pointing device is called anaphora, and words or phrases used in this way are referred to as anaphora..." (Sidner, 1986). For example, SOPHIE resolves anaphors such as:

"What is the voltage there?"

where "there" is an anaphor looking for a pronominal referent, by using the semantic grammar to restrict the class of possible referents. In this example, the anaphor refers to a voltage measurement point in a circuit, where the occurrence of the word "voltage" in the given context triggers the grammar rule corresponding to the concept of measurement. Once the parser has determined the correct class(es) of referent, a history list is invoked which holds all student interactions for the current session, being searched in reverse order when an anaphor (for example, *that, it,* and *one*) is encountered enabling the correct conversational context to be retrieved. An example of anaphor resolution in everyday conversation might be:

9. What is the capital of Scotland?
10. The capital of Scotland is Edinburgh.
11. Is that right?

The resolution of the anaphor "that" in statement 11 refers to the answer given in statement 10.

Deixis

One feature not supported by SOPHIE is deixis. Deixis within an interface, like its natural language counterpart allows users to point whilst talking. That is, they can point to an object and pronominally refer to its parts or whole using deictical or exophoric words, for example, this/that and

here/there (Crystal, 1980). The ability to disambiguate an intended meaning in this way provides a powerful means of reference. An example of deixis used to refer to a hot air balloon might be:

12. Look at that up there. (pointing with a hand)
13. Do you see its basket?
14. Isn't it colourful?

Deixis within direct manipulation interfaces uses the mouse as a pointing device to refer to displayed icons representing domain objects and concepts. The use of deixis within many direct manipulation interfaces, like its natural language counterpart, is ambiguous unless it is combined with some other type of information to decide what is meant. Normally, this information is either spoken or typed natural language.

Given the success of SOPHIE's natural language capabilities, this research seeks to construct a graphical interface which is capable of performing as well as it did by supporting anaphora, ellipsis and extending this to support deixis.

1.2 Research methodology

Five complementary approaches to the research work are explored and applied:

1.2.1 Literature Review

A wide range of graphical interfaces have been developed for Intelligent Tutoring Systems (ITS) used in different domains. A large proportion of these use pop-up or pull-down menus to effect a change in the status of the interface. There also seems to be no standardized convention of interaction when activating combinations of icons to achieve some predefined goal either using menus or icon bars. In some cases, depending upon the convention

supported, highlighted icons corresponding to commands can be followed by arguments, but changing the command portion of the syntax will be taken to mean a new operation. In other interfaces the command portion can be varied but not the argument part. No consideration is given either to the windows displaying separate contexts in these environments. Currently, actions taken in these windows are treated as separate activities from one another, even if they are related.

By contrast, some natural language interfaces support limited surface linguistic features such as anaphora and ellipsis, which allow parts of a sentence to be modified which facilitate the tracking of a conversation as it proceeds. Conversational moves between contexts are supported to some degree as well, albeit limited at present. Although natural language interfaces have a great deal of potential the sheer computational complexity involved in implementing them have motivated other researchers to use menu-driven interfaces. They are just as functional and efficient as natural language interfaces but less complex (Psotka, Massey and Mutter, 1988).

1.2.2 Computational prototype

Given the limitations of natural language interfaces identified by the review and the way they have been successfully resolved to some degree in graphical interfaces, a small prototype system was constructed to see if a small sample of natural language could be modelled graphically.

Circuit I, an object-oriented prototype within the domain of electronics demonstrated that it is possible to express simple dialogue graphically. Within the context of the sample dialogue, questions associated with measurements may be posed and modified using elliptical and anaphoric-like mouse selections.

1.2.3 Pilot study

A pilot study was carried out to establish if graphical ellipsis and anaphoric-like mouse selections modelled by Circuit I performed the same functions as their natural language counterparts. An analysis of users' protocols and their corresponding mouse-clicking actions indicated that graphical anaphoric-like and elliptical equivalents do exist and perform the same functions as in natural language.

1.2.4 Computational implementation

Given the results of the pilot study a more ambitious system, Circuit II, was planned and constructed which extended the range of natural language graphically modelled. The user is presented with a direct manipulation environment which allows them to engage in a meta-dialogue with the system where their intentions are mapped onto the graphical equivalent of circuit components. As well as allowing users to focus their attention on the component, schematic and sub-component levels of the circuit via windows, the system supports surface linguistic phenomena (anaphoric-like, elliptical and deictic mouse selections) but in colour.

A feature of the system is that it supports digressions between three different but related contexts such that, prior to a transition, the current context is saved and is re-instated when it is returned to. Circuit II supports such digressions by providing graphical equivalents of cue phrases, for example, 'Incidentally...' to signal an interruption to a new context and 'As I was saying...' to signal a resumption of a previously suspended one.

Its ability to do this allows users to engage in an extended dialogue where their intentions, expressed as mouse-clicking actions, are interpreted in a way which may well mirror their meaning more closely as they occur at the cognitive level as in natural language dialogue. Circuit II is an attempt to

extend and support coherency between context transitions signalled by the user.

1.2.5 Major study

An observational study with electronic experts was conducted with Circuit II. The findings of the study showed that a direct manipulation interface which supports conversational features, in particular the surface linguistic features and interruption-returns between related contexts described above, supports the notion of conversational coherency.

1.3 Structure of thesis

Chapter 2 reviews the literature on the main interactional styles which have been used through which dialogue between a user and a computer system takes place. The difficulty of constructing a robust natural language interface and how this has led to an increased use of graphical interfaces in ITS is discussed. An in-depth review of the ITS, SOPHIE, is given because it forms one of the core parts of this thesis upon which the two models, Circuit I and Circuit II are based. A range of graphical interfaces used by ITS and the facilities they offer are reviewed as are the limitations associated with them. Literature on innovative approaches which seek to extend the flexibility of graphical interfaces used in ITS are also reviewed. The conclusions of the review are that most graphical interfaces currently being used or developed seem to give little or no consideration to tracking mouse selections made within a context, and certainly not between separate but related contexts.

Chapter 3 provides a full description of the internal and external design of Circuit I, an object-oriented prototype written in HyperCard. Using a simple series-parallel circuit, examples showing how simple objects representing

circuit components and faults can be combined to pose simple and complex questions within a graphical interface are given.

Chapter 4 describes a pilot study carried out to establish if graphical anaphora and ellipsis, as modelled by Circuit I, performed the same functions as they did in SOPHIE's natural language interface. The results of the study showed that natural language ellipsis does have a graphical counterpart which performs the same function. In the case of anaphoric reference, changing the command portion of the syntax performs a similar function to that of natural language anaphora. However, it should be considered anaphoric-like for unlike anaphora in natural language, a search of previous utterances to instantiate a meaning for the pronoun is unnecessary. Despite this difference, these two graphical mechanisms for anaphora and ellipsis provided users of Circuit I with an effective means of posing their questions in a way which was economical in its use of mouse selections.

Chapter 5 details the design of Circuit II which is a re-implementation of SOPHIE I. Descriptions of its parts are given, including the graphical parser, the device specialists, the computational model, how it displays its intelligence and the range of dialogue supported graphically. As well as supporting graphical anaphora and ellipsis as modelled in Circuit I, Circuit II extends this work by a) supporting a more general form of deixis and b) demonstrating that conversational moves between different but related contexts is possible.

Chapter 6 gives a detailed description of the techniques which Circuit II uses to model more complex SOPHIE dialogue.

Chapter 7 reports on the main observational study conducted with Circuit II which gives a comprehensive summary of how users reacted to and benefited from their interaction with the system.

This thesis concludes with **Chapter 8** which summarizes the achievements and main contributions of this research to the area of cognitive science, graphical interface design and semantics. This chapter also discusses the limitations of the current models, Circuit I and II, and considers ways in which this research may be extended for further research and development.

1.4 Chronology of research

There were two major research influences on this thesis: one pragmatic and the other theoretical. From a pragmatic perspective the work on SOPHIE (Brown et al., 1974; Brown, Rubinstein, and Burton 1976; and Brown, Burton, and de Kleer, 1982) has been the major impetus behind the construction of the two models, Circuit I and Circuit II. From a theoretical perspective, the work of Reichman (1981, 1986) on discourse processing has played a prominent role and has been used as a basis to refine SOPHIE's context switching mechanism. The synthesis of these two approaches have been combined to develop a direct manipulation interface to SOPHIE which attempts to a) re-implement its current linguistic performance in a graphical environment and b) extend its conversational capability so that it supports context digressions and resumptions in a more natural way than SOPHIE.

1.4.1 Circuit I

Development of this direct manipulation interface has taken place in two stages. The first problem to overcome during the developmental stage of the model, Circuit I, was to decide which programming language should be used. Two criteria were used to select a programming language. First, it should be easy to construct and control the behaviour of graphical objects. Second, it should facilitate rapid prototyping without the need to get involved in learning complex code. From a wide range of programming languages

available HyperCard was chosen since it fulfilled the set criteria. The model, Circuit I, was constructed very easily, taking approximately twelve weeks to complete, with no major difficulties being encountered. The evaluation of the model was modest but successful. Using only four subjects, the results obtained showed that supporting linguistic mechanisms in a graphical interface was feasible and could offer some advantages to users.

1.4.2 Circuit II

The development of the second model was not so straightforward. Initially HyperCard was used to construct a more complex interface, but this was abandoned because a) the screen was too small to display a reasonably sized circuit, and b) there were many potential pitfalls in developing or obtaining a proper circuit simulator which would either run within HyperCard or could be easily linked to it through XCMDs (external commands). The credibility of the graphical interface depends upon the fact that the circuit modelled by the improved version of Circuit I is of comparable complexity to the Heathkit IP-28 regulated power supply modelled by SOPHIE. To model such a circuit involves the use of a circuit simulator such as SPICE. From this analysis three further criteria were identified in addition to those that governed the design process of Circuit I. First, that the new model, Circuit II, should use a circuit simulator to model a larger circuit of comparable complexity to SOPHIE's. Second, that the computer system used to develop the new model should have a large enough monitor to display the circuit and its associated parts. Third, that the monitor used to display the interface of Circuit II had to support colour. Taking into consideration these criteria and those governing the design of Circuit I, the final choice of system which met all these conditions was the Acorn Archimedes A5000. The implementation language used on the A5000 was BASIC VI which was fast, powerful and ideal for rapid protyping.

The design of Circuit II involved four major stages: a) the design and layout of the graphical interface, b) re-implementing specialists used by SOPHIE, c) modifying the circuit simulator MITEYSPICE and c) implementing parts of Reichman's work on cue phrases as a computational model. By far the most demanding of these tasks was the design of the interface. Given that the interface should respond dynamically to users' actions, (for example, requests for measurements, component modifications) it was essential that the change brought about by users' actions should occur as naturally as possible. A number of different approaches were explored in the graphical environment of Atelier, an advanced development environment. It took a number of attempts before a final version was realized.

Re-implementing eight of SOPHIE's specialists came about by reading the published literature which described their operation. SOPHIE's specialists were all implemented in LISP code, but in Circuit II they are all written in BASIC VI, a highly developed form of BASIC. Since there is no compatibility of coding between LISP and BASIC, all the specialists had to be completely re-constructed. Each specialist uses numerical data output from MITEYSPICE as its input which it then manipulates to provide answers to users' questions. Because of the way the Archimedes stores floating numbers, comparing the results of a calculation against numerical data stored in an array often led to errors. Numerical data held in arrays is stored only to four significant figures, whilst the results of calculations yield nine significant figures. This problem was resolved by reducing the magnitude of each calculation to four significant figures.

Like SOPHIE, Circuit II uses a circuit simulator to model the behaviour of its circuit, an audio video power supply. The simulator, MITEYSPICE, is written specifically to run on an Archimedes system and comes, like SPICE, with its own interface. To use MITEYSPICE in Circuit II involved removing all of the code associated with its interface and rewriting its data output

routines. As a commercial product, MITEYSPICE's program code is not documented, and its modification was non-trivial involving a reasonable understanding of its modules.

Modelling Reichman's work on cue phrases, for example, "Incidentally..."and "Anyway...", also caused some problems as these mechanisms have never been implemented in a graphical environment before. Circuit II is designed to support three different topic shifts, that is, a) normal circuit behaviour, b) a circuit with an unknown fault and c) circuits with user-hypothesised faults. Cue phrases are used to move between contexts a) and c). Deciding how to model these cue phrases was done by exploring a number of graphical alternatives, for example, labels and icons. After many trials the method finally chosen for effecting an automatic move between a) and c) was to use the depression of the second mouse button over a component (indicating that a fault is about to be inserted) or a fault icon. Returning from such a digression is achieved by depressing the first mouse button over a label marked "Norm Circuit." Moving to a context like b) is done by clicking on an icon which represents the insertion of a random fault.

1.5 Summary

The construction of these two models, Circuit I and Circuit II, demonstrates that a graphical interface which supports certain conversational features introduces the notion of conversational coherency within the environment. Features known to support the idea of coherency in natural language systems are anaphora, ellipsis, deixis and cue phrases signalling topic shifts between contexts. The problems of anaphora and deictical references have been resolved in multi-modal systems by specifying the referent in natural language and resolving its meaning using the pointing device. By contrast, merging graphical analogues of such features, and cue phrases for effecting a topic shift within direct manipulation style design, has not

been attempted before. Supporting graphical analogues of these important conversational features provides mechanisms which can track users' mouse selections within and between contexts in a coherent manner. It is suggested that the incorporation of such features within a direct manipulation style design makes an important contribution to the area of interface design.

Related Research

2.1 Introduction

This chapter reviews the literature on some of the main interactional styles that have been developed over the last twenty years. Interactional styles refers to the different ways that users interact with computer systems and include for example, command languages, menus and natural language dialogue. Its purpose is to examine and evaluate the strengths and weaknesses of each style, giving examples of systems which have used them, where appropriate, and to determine how they might be improved upon.

2.2 Command line dialogues

Command line dialogues were one of the first forms of interaction developed for computer systems. These are what Perlman (1984) calls "artificial languages," which he defines as "languages created especially for precise and concise communication within a limited domain." This form of communication uses abbreviated commands which are typed in at a terminal to initiate a task, for example, transferring data from one disc to another.

Research studies on command dialogues (Shneiderman and Mayer, 1979) have shown that the choice of mnemonic names, rather than random ones, make computer programs easier to comprehend. Choosing a meaningful command name is much more subjective because of the wide range of preferences held by people, although there is evidence to suggest that user generated names are slightly better than pre-assigned names (Scapin, 1982; Jones and Landauer, 1985).

In command line systems the inexperienced user has the burden of finding their way through the system, of determining how to go from one mode to another and more importantly of how to return to where they started from. Navigating through such systems can leave them very frustrated and confused because there is no simple way to explore the system and discover the options available (Tesler, 1981).

Despite these disadvantages command line dialogues are extremely popular, particularly so with expert users who through experience of such abbreviated commands can rapidly issue instructions to the computer. The concise nature of command languages makes them very intolerant of input errors but these are easily detected and rectified. Well known computer operating systems which use command line dialogues are DOS, UNIX and CP/M. Norman (1981) has pointed out that the naming conventions used for commands in UNIX make it very difficult to learn. A more recent approach, The Dialogue Development System, has been proposed to extend the flexibility of command languages (Robertson and Burns, 1985).

Dialogue Development System

Research work at the University of Bradford, U.K., has focussed on the Dialogue Development System, a multi-level adaptable system designed to increase the flexibility of command language dialogues.

Designed to be adaptable, a User Interface Specification Language was proposed to a) alleviate programmers from the task of interface management and b) provide more useful feedback when input errors occurred.

The Development Dialogue System is comprised of five main parts: a *terminal database,* a *dialogue manager,* a *validator*, a *screen formatter* and a *system monitor*. All input, in a command language format, is chanelled

through a terminal database which is both device and machine independent. The dialogue manager is responsible for interpreting and responding to all User Interface Specification Language input. A module called the validator acts as a mediator between the dialogue manager and the application software which checks all output to the interface. All output is displayed on the system monitor which also displays menus via the screen formatter to answer follow up queries to be made. The system is capable of taking the initiative during the interaction by prompting users for missing arguments to input commands.

2.3 Natural language interfaces

Natural language processing research originated back in 1956 at the Dartmouth Conference on Artificial Intelligence (Grosz, Jones and Webber, 1986). As an interactional style it has much to offer, not least, expressiveness of user input and that it provides a means of access to the system which is natural without having to learn a new language. However, to construct an effective natural language interface is difficult because it requires large amounts of linguistic and world knowledge (Petrick, 1976, Rich, 1984). Rich (1984) points out that a system which can only understand a small and ill-defined subset of English will be ineffective as a natural language interface. She identifies three major factors which contribute to the difficulty of understanding natural language input:

- *The complexity of the target representation into which the matching is being done.*

For example, the translation of natural language input in a keyword based data retrieval system to extract information is not so complex as when it is used to describe and record events and their relationships.

• *The type of mapping: one-one, many-one, one-many, or many-many.* Mapping a natural language statement into a form which can be understood and used by the computer is not easy. One to one mappings are by far the easiest to cope with and at the other extreme, one to many mappings require a great deal of world knowledge which is often not available to the computer.

• *The level of interaction of the components of the source representation.* A natural language sentence is composed of many parts, for example, nouns, verbs, adjectives and so on. When a sentence is broken down into its constituent parts during the parsing process a structure of that sentence is formed. Changing a word within that sentence can have the effect of altering its parsed structure and thus its meaning.

Despite the difficulties involved in implementing a natural language interface many have been constructed as a communication medium to computer systems. Three basic techniques to-date have been used to construct computer models of natural language interfaces, that is, *keyword matching*, *syntactic analysis* and *semantic analysis*.

2.3.1 Keyword matching

The most well known keyword matching system was ELIZA (Weizenbaum, 1966) which simulated the behaviour of a Rogerian therapist. This system analyzed all sentences input into it by matching them against a template. The advantage of this technique is that it allows sentences whose grammar is unusual or ungrammatical to be recognized. The disadvantage though is that it disregards a great deal of information which the sentence may contain (Rich, 1983). Other keyword matching systems include the Intelligent Tutoring Systems (ITS) SCHOLAR (Carbonell, 1970) and METEOROLOGY (Brown, Burton, and Zdydel, 1973).

SCHOLAR

SCHOLAR was an ITS which could conduct a mixed-initiative dialogue with a student. Its interface used a limited set of English to generate simple sentences and questions through templates that were filled in with information from a semantic net holding domain knowledge. Student-posed questions were parsed by breaking down the sentence into domain concepts which were then used to traverse the semantic network. Information retrieved was then used to fill in the sentence template which was used to respond to the student's question. This is in contrast to student answers which were scanned for keywords. These were compared against those produced for the answer by the semantic net which, if correct, signified that the student had answered correctly. Although SCHOLAR recognized wrong answers it could not diagnose students' misconceptions. However, this problem was re-addressed in a later version of SCHOLAR (Collins, Warnock, Aiello, and Miller, 1975) by adding a module which, when it detected errors in answers given by students, could provide them with a report of factors which distinguished their answer from the correct one.

2.3.2 Syntactic parsing

Syntactic analysis, by contrast, captures the richness of information contained within a sentence, unlike keyword matching, by breaking down (parsing) the input sentence into its constituent parts, for example, nouns and verbs. To do this a grammar is used which describes the rules of the language so that the structure of a sentence can be formed during the parsing process called a parse tree. One of the most successful parsing strategies yet developed is the Augmented Transition Network grammar (Woods, 1970), which recognizes sentences efficiently by incorporating a wide variety of knowledge into the parsing system. This approach has a number of disadvantages: a great deal of backtracking (reversing decisions during

the construction of the parsing tree) may be required if a sentence is ambiguous. If a sentence is ambigous there is no way that heuristic functions can be applied to decide its correct meaning. Should the sentence contain words unknown to the system and the parse tree path not match exactly a path in the network the parsing process will fail. These drawbacks aside, the ATN remains a very useful mechanism and has been used in systems like LUNAR (Woods, 1973, 1977) and ROBOT (Harris, 1977).

LUNAR

The LUNAR system was an experimental natural language interface to a database of lunar rock samples brought back by the Apollo 11 mission. It was designed to assist lunar geologists to analyze data pertaining to the samples.

The system used an ATN parser, a semantic interpreter which employed a complex parsing process. The parsing process started by carrying out a syntactic analysis of the input sentence that produced a parse tree which was analyzed by procedures. These procedures determined whether the sentence was declarative, imperative or interrogative. The semantic interpreter then converted the input into predicate calculus like statements each representing a *Contingent Knowledge Structure* (CKS). Each CKS, representing a synthesis of its raw input was used itself as input to the next stage of the process. Next the analysis phase was carried out by LUNAR's interpreter which extracted information from the fixed database by executing the CKS as a program to provide answers to questions.

2.3.3 Semantic grammars

Semantic grammars (Burton, 1975, 1976) provides what Wenger (1987) rightly calls "a desirable middle path" between keyword matching and syntactic parsing. A semantic grammar is a context free grammar which

decomposes a sentence into meaningful categories which are domain specific. Regular grammars (i.e., syntactic analysis) by contrast decompose them by syntactic categories i.e., noun and verb phrases. The advantage of semantic grammars is that they interpret the meaning of a sentence during the parsing process, unlike syntactic grammars, thus allowing the result to be used without further processing. Also, ambiguities which can arise during a strict syntactic parse can be avoided because the sentence is analyzed by semantic categories. The major drawback of this approach is that, since many generalizations are not present in semantic grammars, rules for each case have to appear more than once. Depending upon the size of the system the number of rules required can be prohibitive thus reducing the efficiency of the parsing process. Despite these drawbacks semantic grammars have been used effectively in many systems, including SOPHIE (Brown, Burton et al., 1974, 1976 and 1982), LIFER (Hendrix et al., 1978), PLANES (Waltz and Goodman, 1977) and ACE (Sleeman and Hendley, 1979).

SOPHIE

The construction of SOPHIE I (Brown et al., 1974) was an attempt to extend the work of SCHOLAR, an ITS which used keyword matching parsing techniques. SOPHIE, like SCHOLAR, is considered a classic in the field of ITS which has motivated the work presented in this thesis. Since the computational capabilities of SOPHIE have been re-implemented in Circuit II (see in §5.0 and §6.0), this section reviews SOPHIE's capabilities.

SOPHIE (SOPHisticated Instructional Environment) is a reactive learning environment for debugging electronic circuits which tutors students on how to troubleshoot. Rather than providing step by step instructions on how to debug an electronic circuit, SOPHIE provides students with an "expert" which helps them to formulate and debug their own ideas. Meaningful feedback is given to student-posed questions and hypotheses by a series of

inference specialists which analyze the results produced by the circuit simulator SPICE (Nagel, 1971; Nagel and Pederson 1973) when invoked. In this environment students may explore the limits of their theoretical knowledge of electronics and gain a deeper appreciation of causality underlying circuit devices.

SOPHIE was the first to use semantic grammars to construct a powerful natural language interface. Four criteria motivated the design of the interface (Burton et al., 1979): 1) parsing keyed input should be done as efficiently as possible so as to ensure the continuity of students' thought processes, 2) the interface should be able to accommodate the wide variety of ways that a student may express herself, 3) the system should be robust enough to handle incomplete and ambiguous sentences as a student's familiarity with the system increases, and 4) help facilities should be provided to explain the range of natural language input handled by the interface.

The semantic grammar used by SOPHIE, which decomposes a sentence into meaningful categories, provides its parsing mechanisms with well defined and constrained semantic information about the domain. This in turn enables the parser to accommodate the wide range of ways that students may express themselves. Students' questions and answers, expressed in terms of these categories, are then used by the parser to interpret the meaning of input sentences. When the parser comes across incomplete or ambiguous sentences, rather than rejecting the sentence as ungrammatical, it has the ability to predict what its correct meaning is. This is particularly so in the case of anaphoric and elliptic inputs whose meanings are resolved by searching a history list of previous utterances backwards. This approach gave SOPHIE a very robust natural language interface which, by its natural responses, gave users the impression that it was very knowledgeable.

Most of SOPHIE I's intelligence was manifest through its clever use of procedural specialists, each with a specific function, which collectively supported a laboratory environment within which experiments could be conducted. The functions of each of these specialists are described below.

The state of the Heathkit IP-28 regulated power supply was determined by the control settings, its load resistance and the faults which were inserted into the device. Context information was held and maintained by two specialists, i.e., the 'setting' and 'fault' specialists. If the control settings and/or the type of fault was altered then these specialists were invoked and the information held by them updated.

SOPHIE provided students with four measurement specialists which allowed them to determine the state of the device. The 'voltage' specialist was used to determine the voltage across the circuit's components or inspection points. By accessing the semantic network, which described the structure of the device, information was obtained which enabled the voltage table to be referenced for a component's specific value. The other specialists, i.e., current, resistance and power were used in a similar manner.

Within a random fault context students were free to use their troubleshooting skills to isolate the problem. In this mode they invoked the 'fault questioning' specialist by asking if a particular component was faulted or not. This specialist first checked to see if a) the component was faulted and b) it was faulted in the manner suggested by the student.

Also, within a random or hypothetical fault context, students took measurements which they needed to check to see if they were a symptom of the fault. To do this they invoked the 'measurement checking' specialist by typing "Is that right?" which determined the correctness of the answer by comparing it to the same measurement made in an unfaulted context. The

results of this comparison were then given to the student by displaying the normal working measurement against the observed measurement.

When troubleshooting a circuit a student will form several hypotheses about the nature of the fault which they will wish to explore. SOPHIE supported this type of exploration through the 'hypothesis testing' specialist. After taking a number of measurements in a random fault context students could present their hypothesis to the system by typing for example, "I think that the base-emitter of TR1 is open". Once given, the circuit simulator was modified to reflect the suggested fault and re-run to obtain new values for this specific fault. Each measurement taken prior to the hypothesis being posed was re-taken and compared to the observed value. If all the measurements taken were the same then the student's hypothesis was consistent with what they had observed. Measurements that differed from the observed ones were brought to the student's attention and reflected either a misunderstanding of the circuit's operation or a deficiency in the their reasoning processes.

Another specialist supported by SOPHIE was the 'conditional' specialist which answered "If... then..." types of questions. Students invoked this specialist when they wished to explore hypothetical scenarios for example, "If C2 shorts what happens to the current through the collector of Q5?". In this example the specialist would modify the circuit model to reflect the fault, that is, C2 would be shorted and the circuit simulator would be invoked to derive new values for this fault. Once computed the measurement specialist would be invoked to determine the current through the collector of Q5. Both the faulted and working measurements would then be displayed to the student.

After taking a series of measurements, if a student was unable to suggest a fault which could explain their observed measurements, they could ask for

assistance by invoking the 'hypothesis generation' specialist. Based on the measurements taken to date, this specialist would determine the set of possible faults that could account for these measurements. This was, and still is, a non-trivial task. Starting with the last measurement taken and working in reverse, a 'proposer' specialist proposed a list of possible hypotheses which might account for that measurement. Each hypothesis generated was evaluated by the 'refiner' specialist which set up and ran the circuit simulator which compared all measurements taken to verify or eliminate the hypothesis. An 'instantiator' specialist, in cases where hypotheses were not fully specified (for example, "the beta of TR1 is low") was invoked to instantiate a value which might account for it. Because this is an iterative process, a special functional simulator more efficient than the general purpose circuit simulator was used to derive this value. The hypotheses generated were all tested until only those hypotheses which could explain the observed measurements were left.

The most complex task performed by SOPHIE in a random fault context was to evaluate a student's most recent measurement to determine if it added any new information as to what the nature of the fault might be. Using the hypothesis generation specialist, two lists of hypotheses would be generated, those consistent with the new measurement and those consistent with the previous measurement. If the set of hypotheses generated were the same then the new measurement was considered redundant, and if not then it shed new light on the nature of the fault.

The seat of SOPHIE's intelligence lies in its use of specialists which analyse and deduce information derived from the data output by its circuit simulator SPICE. The delivery of this information to students using a natural language medium enables SOPHIE I to support a powerful instructional environment. SOPHIE II, a successor to SOPHIE I extended this environment by supporting an expert module capable of troubleshooting a

faulted circuit. The instructional advantage of this extension to students was that the expert could demonstrate a range of effective troubleshooting strategies to locate the fault. SOPHIE I and SOPHIE II were limited because they calculated their answers in a systematic way by running the circuit simulator under various conditions. This meant that, although they could deal with student questions and hypotheses they could not provide a justification of their reasoning for arriving at their answers. This deficiency, in the quantitative simulation environment, to provide causal explanations for circuit behaviour was noted by Brown et al. and spawned a new system SOPHIE III. SOPHIE III was a major departure from SOPHIE I in that at its core was a causal, rather than a mathematical, model from which its electronic expert, troubleshooter and coach modules derived information.

The image of SOPHIE as a seemingly knowledgable and intelligent tutor of electronics was well founded. However, the success of SOPHIE's natural language interface and the intelligence it exhibited through its inference capabilities frequently led users to adopt a vocabulary which exceeded its ability to respond correctly. During such exchanges SOPHIE handled limited forms of anaphora, ellipsis and conversational moves but within human dialogue a whole range of such forms can be expressed. Supporting these other forms in natural language systems is very hard and still remains a research issue. Whilst using semantic grammar enables a natural language interface to be constructed quite quickly, the technique of using domain concepts greatly restricts the parts of it that can be reused building another interface for a new domain.

2.4 Menu dialogue

Given the inherent difficulties of using natural language as a communication medium, over the last ten years attention has increasingly

focussed on graphical interfaces. During this time a plethora of menu-based systems have been developed. They seem to provide users with a more effective means of communicating their intentions to the system than is currently possible with natural language systems. Instead of typing long sentences users simply point to and select items from displayed menus. With the rapid growth of menu-based interfaces came the need to examine the pertinent features concerning menu system design. Shneiderman (1986) has conducted a comprehensive review of previous research work which has examined these issues, for example, whether or not menu systems should have deep or shallow tree structures.

One view of menu driven interfaces is that they provide inexperienced users with a means through which they can navigate the system as well as learning about the range of commands that it can perform (Burton et al., 1979; Norman, 1983; Perlman, 1985). Another view is that they are cumbersome to experienced users who merely wish to execute a command without having to navigate a range of menus (Perlman, 1985). Two further points are that the menus take up too much screen space and that there can be a delay plotting them on computer systems with slow processors. Norman (1983) has shown how these two points influence the design of an interface. Computer systems with slow processors and small workspaces bias the design towards command language systems. By contrast, fast computers with large work spaces favour menu-based systems. Menu driven software is widely used within a range of different application programs, for example, Borland's Turbo Pascal, Prolog, C, and BASIC. Examples of specialized systems which have used it as their only means of interaction are a) PROMIS (Schultz and Davis, 1979; Schultz, 1986) a medical information system and b) ZOG (Robertson, McCracken and Newell, 1981; McCracken and Akscyn, 1984) a rapid-response, large-network menu selection system.

ZOG

Since its implementation, ZOG has been used as an interface for a command language system, a database retrieval system, a Computer Assisted Instruction system, a guidance system, an interrogation system and a question-answering system. In its most recent form, a distributed hypermedia system, called Knowledge Management Systems (Akscyn and McCracken, 1988) runs on Apollo and Sun workstations.

ZOG uses a hierarchy of subnets, each being a tree of frames organized in a database form which can be displayed to the user. Each frame contains information pertaining to the context within which it is used and is identified by a unique number and title. Traversing the network is achieved by selecting available options, by touching sensitive parts of the screen or by single character key presses, for example, to move to another frame. In this way users can navigate the system collating information and, when required, can review a list of frames visited if desired.

ZOG provides the system developer with a frame editor, ZED, which can be used at any time. Users can also use the editor to personalise their frames if they wish. In a similar way to HyperCard scripts, frames hold hidden information about the name of their creators and access level of its current users.

Frames also include 'action text', which when selected is sent to some other destinations via a communications port to perform some type of action. In addition, frames in one implementation of ZOG can be converted to an external format and transferred to other implementations thus making it very portable.

In use ZOG suffers from a number of problems as with other hierarchical menu-driven systems. Users can get lost whilst navigating the system and

the size of the screen display also has been found to impose a heavy burden on users' short-term memory (Robertson et al., 1981). This problem was rectified in the commercial version of ZOG, Knowledge Management Systems, by allowing users access to the previous and current frames simultaneously.

2.5 Direct Manipulation interfaces

The design of an interface may be considered an iterative process within which interactive styles, ideas and concepts are explored and developed. During the evolutionary process of interface design one such interactive style which has enjoyed increasing popularity is *direct manipulation*. This style of interaction gives users the feeling of directly interacting with the domain and is another approach which seeks to overcome the limitations of the previous techniques discussed.

Direct Manipulation interfaces first came about with the development of the Xerox Star (Smith, Irby, Kimball, Verplank, and Harslem, 1982) and the Apple Macintosh iconic interfaces. These interfaces have three distinguishing features which are: a) the object of interest must always be visible, b) the object of interest must be directly manipulated, i.e., not by a command syntax and c) it must be possible to perform rapid incremental reversible operations on an object of interest which is immediately visible (Shneiderman, 1982).

Central to the direct manipulation technique is the icon or object of interest, a pictorial representation of an idea, object or concept, for example, a file, a waste-paper basket, programs or data file. The function of an icon, selected with a pointing device, need not be remembered as its representation should indicate its function.

The inclusion and use of icons within graphical interfaces has caused mixed reactions among researchers. For example, Lodding (1983) claims that they can reduce the time it takes to learn the system and enhance user performance. Gittens (1986) asserts that they will enhance and extend the communication bandwidth between user and system, but notes how difficult it might be to find an appropriate icon representation for particular domain concepts or systems with many command options. Arguments against icon representations include those made by Koler (1969), who claims that a significant amount of intelligence, as well as perceptual and abstraction skills, are required to derive their meaning. Manes (1985) contends that the meaning of an icon can be confusing and inefficient in terms of its use of screen space. Jervell and Olsen (1985) also point out that the meaning of nebulous system concepts represented as icons can themselves be difficult to interpret thus offering no real benefit over other approaches. Icons are multifaceted in that they can have more than one meaning. Assuming the icon is designed correctly, its intended meaning will depend upon the type of mapping convention supported as well as the user's knowledge of the chosen domain and the context in which it is viewed. Rogers (1989) describes two types of mapping which can be used to convey the meaning of icons, namely 'concrete objects' in association with 'abstract' symbols and the use of analogy. The results of an experiment she conducted in a word processing environment found that the meaning and operations of icons were more apparent and memorable when concrete objects were associated with abstract symbols. By contrast, the meaning of icons conveyed by analogy were more difficult to learn and were only memorable when the link was bizarre. Currently, iconic representations, unlike a spoken language, lack any kind of syntactic and semantic rules which can be used to resolve the intended meaning of actions within a graphical interface.

As has been already discussed, icons can have more than one meaning depending upon the context they are viewed in. Rogers (1989) points out that the multi-dimensional nature of icons can be exploited by mapping the contextual meaning of the icon onto the structure of a command set which can be used to perform operations on objects. This provides users with a form of graphical grammar, albeit simple, which makes it easier to learn the range of commands offered by the system (Gittens, 1986).

Whilst the issues concerning the graphical representation of icons are important they apply more to graphical than direct manipulation interfaces. In direct manipulation interfaces the focus of attention is much more on the usability of the interface and the directness of the interaction experienced by users. Direct manipulation and graphical interfaces are also logically separable. For example, you could have a direct manipulation interface system where all of the objects were represented as text labels rather than icons. Within direct manipulation interfaces which use icons, these are directly visible and as Shneiderman (1982) has pointed out, it is the ability to receive immediate visual feedback of performed actions and the results of these actions which are important.

A current limitation of direct manipulation interfaces, and in graphic interfaces in general, is that there exists no general theory for directing the effective use of sophisticated graphics techniques (Miller, 1988). Most graphical interfaces at present employ rigid forms of syntax which can lead to unnecessary work for users.

Popular graphical interfaces currently used only support either an argument-command or command-argument syntax. For example, MacDraw supports the argument-command syntax as shown in figure 2.1a, where only the command portion of the syntax may be varied. The argument 'square', which has been previously selected has been used with

the command 'fill with black' to change the colour of the square from white to black. Selecting a circle (argument) in favour of the square does not change the colour of the circle from white to black (see figure 2.1b). MacDraw also supports the command-argument syntax for other functions.

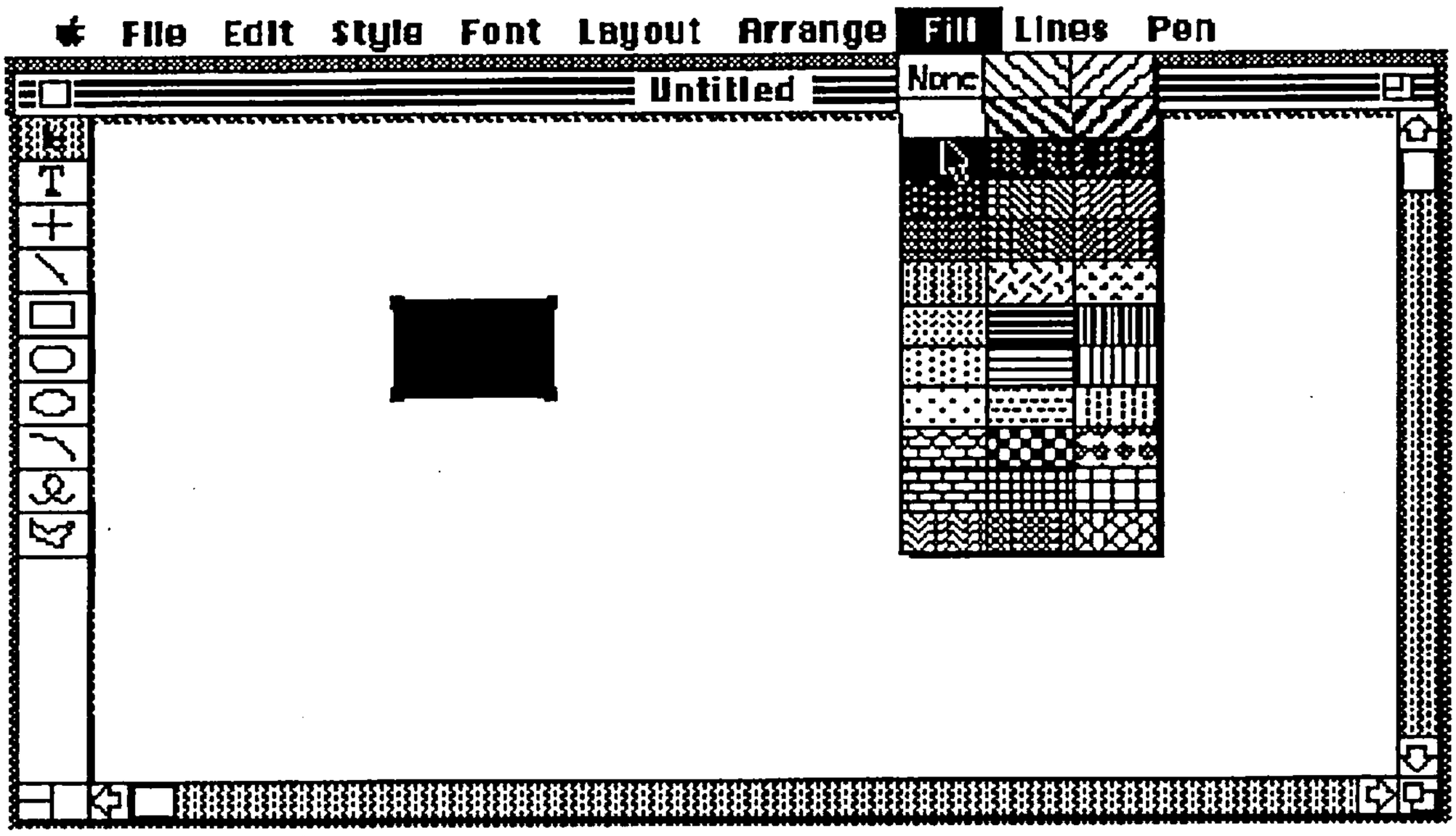


Figure 2.1a MacDraw's argument-command syntax

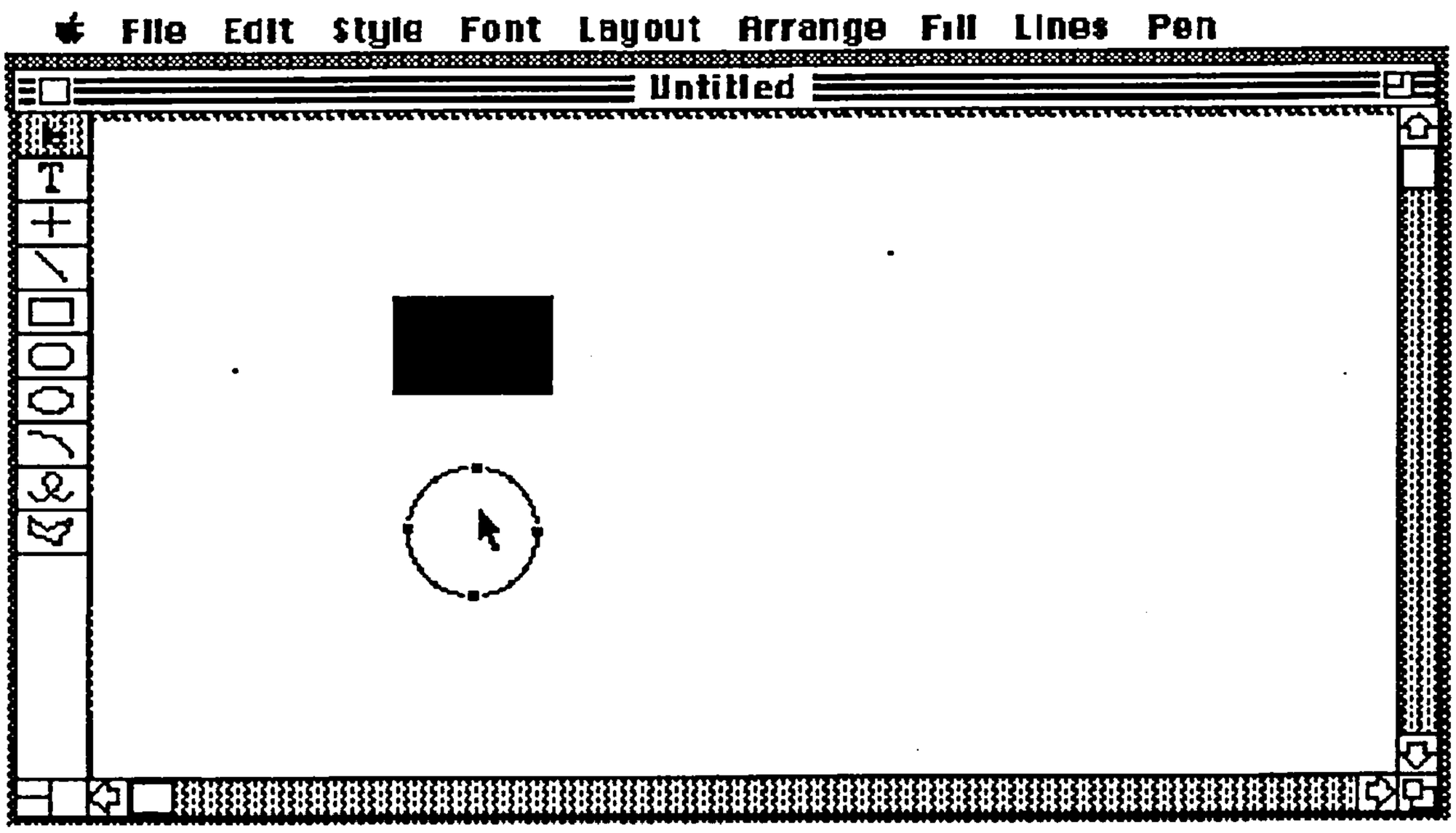


Figure 2.1b MacDraw: no support for command-argument syntax

However, both forms of the syntax are not available to the same function which limits its usefulness.

This is in contrast to MacPaint which supports the command-argument syntax, where, for example, a chosen pattern may be applied to a number of different objects as shown in (a), (b) and (c) of figure 2.2. Attempting to vary

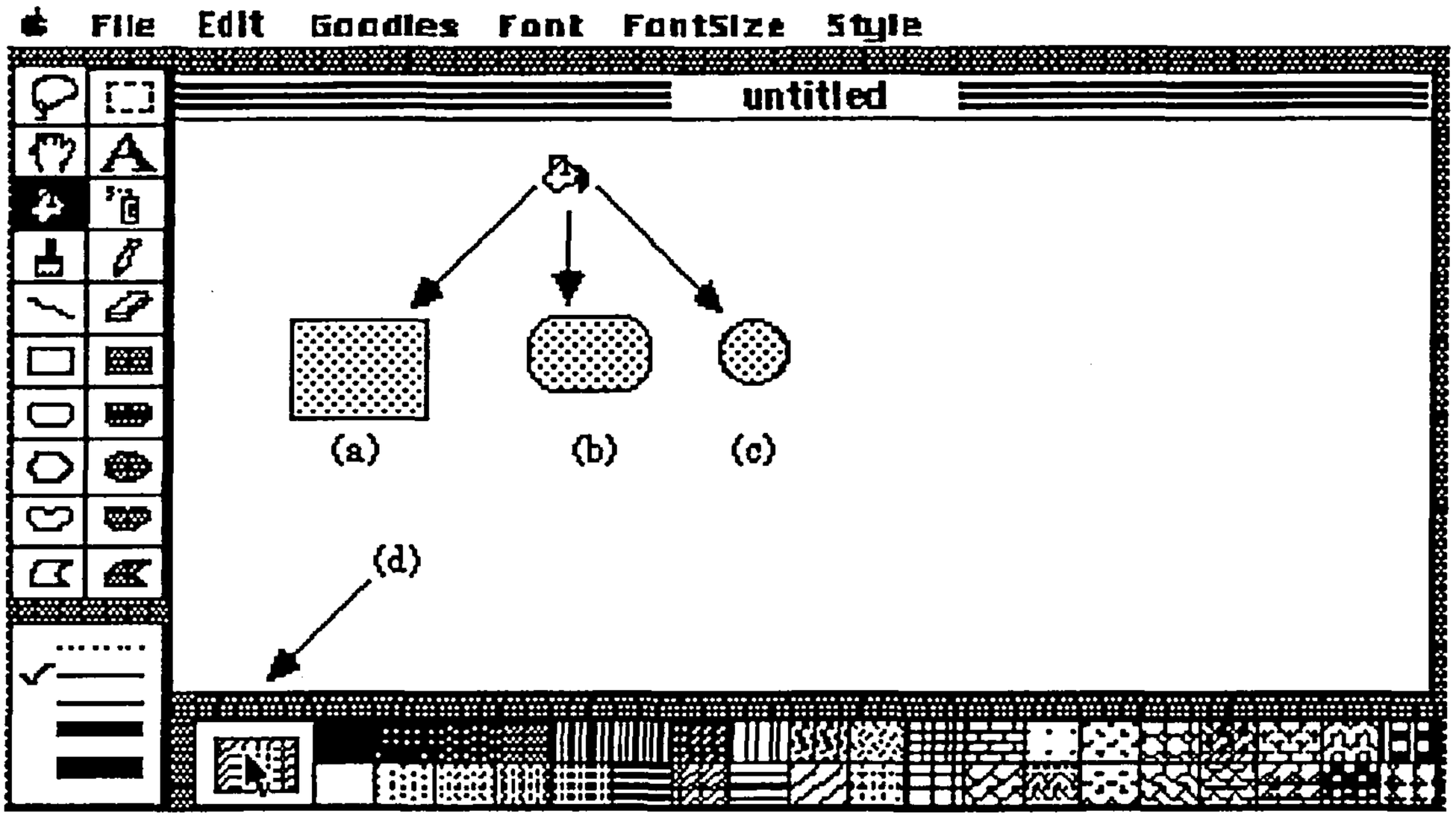


Figure 2.2 Examples of MacPaint's command-argument syntax

the current command in favour of another (see (d) in figure 2.2) has no effect on the last argument (circle) referred to.

Another popular Macintosh application, SuperPaint, provides 'technical drawing' and 'paint' environments which users can move between. Although the paint and drawing environments support the command-argument and argument-command syntaxes respectively as described for MacDraw and MacPaint above, they cannot be used in conjunction with one another. This limited support of syntax restricts users' ability to express themselves in ways which closely parallel natural language expressions.

Direct manipulation interfaces generally support a command-argument+argument (elliptical) syntax which can be expressed either

directly on the objects of interest or through the use of pop-up or pull-down menus. An elliptical sequence of repeated operations expressed without reference to menus is more economical in terms of mouse selections made than when menus are referred to.

Direct manipulation environments, such as pop-up or pull-down menus, can be configured using the options available to set up a particular working context. When a digression to another context occurs however the environments usually cannot store the settings of the previous context which means that when it is returned to it has to be set up again. Another desirable feature often not supported within such environments is the ability to compare and contrast, for example, a measurement taken in one context against the same measurement in a different context. It is suggested that addressing these issues would enable the graphical environment to become more conversational and so allow users a greater degree of expression than is available at the moment.

Many claims are made in support of direct manipulation interfaces although as yet the theory behind the technique is not well understood (Hutchins, Hollans, and Norman, 1986). An example of a system which uses direct manipulation techniques is STEAMER (Hollan, Hutchins and Weitzman, 1984).

STEAMER

STEAMER is an instructional tool for training marine engineers which uses direct manipulation techniques. Its graphical display allowed users to view and interact with a simulation model of a naval steam plant through mouse selections.

Icons representing its different parts, for example, pipes, gauges, valves, could be directly manipulated by the student to control the simulated plant,

such that the flow of water through a gauge could be controlled by directly clicking on it with the mouse. To users inexperienced with STEAMER this was not obvious which was contrary to the nature of direct manipulation interfaces (Miller, 1988).

It was claimed that by manipulating the simulation's parts users could observe the structural relations between different parts of the simulation and acquire a *mental* model of the plant's operation. However, as STEAMER only used an underlying quantitative model to update its graphical display it cannot be said to impart a mental model of the steam plant to the user. The simulation only reflects an abstract view of the computational model (Wenger, 1987).

A graphic editor is also supported by STEAMER so that training instructors can modify or extend the graphical simulation. Within this editor, icons representing STEAMER's various parts can be constructed or manipulated and linked to variables associated with the underlying computational model.

2.6 Windows

Another prominent development in interface design has been the advent of windowing systems. Window systems were first implemented on a Xerox Alto (Thacker, McCreight, Lampson, Spraill, and Boggs, 1982), a research system which was then later developed for commercial purposes on the now famous Xerox Star workstation (see §2.5). The forerunner of all modern window systems, such as the Apple Macintosh, the Star utilized many of the concepts now used in direct manipulation interfaces, for example, icons, windows, and WYSIWYG (What You See Is What You Get).

Windowing systems comprise three parts which are layered and which interact with one another (Jones and Downton, 1991). At the top level are user *interface tools* which imbue an interface with a particular look and feel.

At the second level is the *window system* which controls the creation and removal of windows as well as mouse and keyboard cursor positions. At the third level is the *imaging model* which receives all input commands from levels one and two and translates these into suitable commands which can be executed on the operating system.

For the display of window systems to function satisfactorily both the window control and user input must be properly supervised. This supervision is carried out by *display* and *event* managers. Basically, the display manager is responsible for controlling the display of windows of variable and varying size and shape. It also supports window manipulation by the user, allowing them to move, activate, create or remove windows in the display (Algers, Benyon, Davies, Dobson, Head, Keller, Passey, Preece, and Rogers, 1990). All pointing device or keyboard input is controlled and translated into operating system commands by the event manager. The usability of the window system depends on the cooperative action between display and event managers (Jones and Downton, 1991).

Since their introduction, window systems have gone through various stages of development and can be classified by window implementations, for example, scrollable, frame-at-a-time, split, tiled, overlapping and pop-up. The choice of implementation used during the design process should be governed by the types of tasks users will perform at the interface (Card, Pavel and Farrel, 1985). They also point out that the power of window systems lies in their ability to have a dual function by acting both as a communications medium and as an extension to users' own internal memory by displaying and keeping track of information. The promise of the window paradigm is that the display of independent but related objects of memory, which have links with other objects in the same and/or different contexts, offers the possibility of enhancing human-computer interaction.

The strong link that exists between graphical interfaces and human cognition suggests that the design and interactional techniques in such environments should be guided and informed by how users work (Cypher, 1986; Miyata and Norman, 1986; and Reichman, 1986).

Cypher points out that users do not perform one task at a time, rather they structure their tasks into a linearized sequence to carry out multiple activities, each of which compete for attention. Computer systems, he says, should be "designed so that they actively support and facilitate multiple activities...". Consideration should be given for how activities are related and, where they are, this information should be displayed simultaneously on the screen. Visual and contextual cues should also be provided for interrupted activities since it is generally the case that more information is required to resume an interrupted context than when you are actively performing it (Cypher, 1986).

By contrast, Miyata and Norman (1986) focus on cognitive processes used during an interaction, in particular those used to support multiple activities. As interruptions of current activities occur at unpredictable times, they suggest that computer interfaces should be designed so that the suspension of an activity is made easy. Where the suspension does not require much activity then the context of the current task and the memory of the interrupting task should be preserved by the user (Miyata and Norman, 1986). The interrupted context should also be saved by the computer so that when returned to, the task may be resumed from the position where it was interrupted. They also suggest that the system should provide memorable "reminders" of activities which are suspended and those which should be resumed. Such reminders would enable some of the limitations of human memory and information processing capabilities to be overcome.

Window systems don't at present support multiple activities very well, although the Rooms system (Card and Henderson, 1987), a window management scheme, tries to improve upon this by supporting task switching between activities. The Rooms system when started displays a series of individual Rooms (windows) on the screen each of which represent tasks, for example, reading mail, and word-processing. Navigation between Rooms is achieved by selecting icons representing 'Doors' which give the user "the illusion of transiting to a new Room, containing other windows." Upon entering into a Room the system creates a new Door (in reverse video), called a 'Back Door' through which the suspended task may be resumed. To support user orientation as the number of Rooms grow, the system provides a number of forms of help, including a) a pop-up menu with the names of all currently created Rooms, b) a pictorial overview of the set of Rooms which exist, c) expansion of window pictograms to assist in identifying or searching for particular windows, and d) showing the extent of connectivity which exist between different Rooms. These navigation and orientation aids together with multiple cues, for example, size, shape and arrangement of windows facilitate the exploration of the Rooms system.

Of particular interest is Reichman's (1986) work which suggests a communication paradigm for window systems which might better support multiple window interactivity and communication. Drawing on her previous work on conversational coherency in natural language systems Reichman (1978a, 1985) considers how the conversational metaphor might be supported in window systems. To place her communication paradigm for window systems in context, an overview of her work, conversational coherency in natural language processing, is now presented.

Based on Grosz's (1977) research into focus spaces in task-oriented dialogues, Reichman's (1978a) model of conversation describes how a natural language interface should be able to support the notion of

conversational coherency. To do this the interface must be able to construct a discourse model which breaks a conversational exchange into what Reichman calls 'context spaces.' Once introduced, it must also be able to focus upon the topic of a conversation as it progresses via the use of surface features such as deictics, anaphora and ellipsis without having to continually re-introduce it. In order to understand a conversation listeners will form a hierarchy of related context spaces to form an integrated view of it. Context spaces can be distinguished according to their type which Reichman describes, for example, 'issue' and 'event'. Constituents (i.e., actors, objects, events/issues, location, time, and duration period) within a context space can also be assigned different focus levels, (i.e., high, medium, low, and zero) which, from a speaker's perspective, is one of the ways used to indicate the topic of a conversation. For example, excerpt 1 below illustrates how the issue, "His test equipment" (statement 1), which is initially in low focus, shifts to high focus, "it" (statement 3) then to medium focus, "the oscilloscope" (statement 4) and then again to high focus, "it" (statement 5).

Excerpt 1

K: 1. What happened? *His test equipment*
2. has just failed. ["His" refers to John an electrical technician]
3. One minute *it* was working and then all of a sudden there was
4. a blue flash and a bang. He only bought *the oscilloscope* last
5. month. *It* will cost a fortune to get repaired.

When a context space is closed all constituents within that context space will be assigned zero focus levels. These rules attempt to emulate the way in which humans assign focus levels to various utterances in conversational exchanges.

As well as partitioning the conversation into context spaces and determining the focus levels of their constituents, speakers use cue phrases, for example, 'but', and 'anyway', (Reichman, 1981) to effect a 'conversational move'.

Conversational moves represent the set of relationships which hold between constituent context spaces, a taxonomy of which is given in Reichman, (1978a). The control of these context spaces is achieved through the application of fifteen Semantic Relational Rules which determine the coherence of the conversation (Reichman, 1978b).

The Augmented Transition Network (ATN) (Reichman, 1984) which she discusses as a theoretical model is proposed as a mechanism through which conversational moves can be effected. Reichman claims that her theoretical ATN model, with minor adaptations, could be used as a computer program capable of behaving as a conversational participant.

Reichman's work has been critiqued by Delin (1986), who states that, even if the ATN model were implemented, minor adaptations to it would be insufficient. Rather it would have to be extended to incorporate an additional 'hearer model' of the discourse and a comprehensive discourse repair model. The most serious criticism made by Delin is that Reichman claims to have isolated rules that apply in any sort of discourse when in fact all of Reichman's data is composed of conversations between equals. Delin applied a part of Reichman's theory to some question/answer data but found difficulty with the context space categorizations. More recently, Blandford (1991) has stated that it is not clear how this network can be used to generate discourse moves, and as such must remain theoretical.

Reichman (1986) examines the desktop metaphor which underlies many windowing systems and notes that in such systems windows are usually treated as separate processes which conflicts with the way users view their interaction. Users, she argues, see their actions as being purposeful and coherent, rather than separate and random which is how most window systems presently treat them.

Translating her ideas from discourse processing to windowing systems, Reichman maintains that the user interaction taking place between windows should be tracked forming a discourse model of the interaction. To remedy this shortcoming of current window systems Reichman suggests the development of a visual constraint language, which reflects the interrelationships between and within the different contexts. As a possible contender the constraint language used in spreadsheets is put forward as a means of contextualized navigation through related window activities.

Reichman also compares certain mouse activities in a Lisp window environment to pronominal and deictic reference found in everyday discourse. Consider two windows, one a window for editing Lisp programs and another for running and displaying the results of program execution. When the Lisp program is executed and, say, an error is flagged, moving back to the Lisp environment without clicking to re-activate the editor window is, states Reichman, equivalent to using a pronominal or deictic reference. She argues that the execution of the program does not close the editor process, that there is an expectation that this context will be resumed again. However, in this case, mouse movement (without clicking) is insufficient to activate the editor context again. If, on the other hand, the editor window context was closed then a mouse click over, for example, a pop-up menu, would be appropriate to create a new window. Closed window contexts should be removed suggests Reichman. This type of interruption/resumption and creation of a new window clearly has parallels with Reichman's context space theory already discussed.

Just as differentiating the status of different context spaces in everyday discourse is important, Reichman argues that it is also important to reflect this within the different contexts represented in the window environment. For example, when returning to the Lisp Editor after an error in the Lisp environment, the Lisp environment, in Reichman terminology, is in

'controlling' status. The reason for this is that the actions performed next by the user will be controlled by what happens in the Lisp environment. This is also true of user actions in the Editor window. Again, like windows with a closed status, open window contexts which have been interrupted should, says Reichman, be removed from the screen because they do not control the users' actions. This is in contrast to controlling window contexts which do affect what actions users will perform next and thus should be left displayed. On the whole, window systems which employ the desktop metaphor do not support these types of constraints.

Reichman then goes on to compare and contrast the similarities between the conversational metaphor and the assistant metaphor which she argues perform similar functions. The advantage of assistants is that they are more tolerant of errors than a second conversational participant, and so the assistant metaphor might be more appropriate in controlling window contexts.

In supporting multi-window interaction, Reichman proposes using colour markers to reflect the status of window's contexts and the relationship between them, particularly for sets of related windows.

Despite the criticisms made of Reichman's theory of discourse for natural language processing, her work, applied to windowing systems, has much merit which deserves consideration. Reichman has shown how window systems can be adapted to support multiple activities and perform analogous conversational actions for example, cue phrases, to move between related window activities. She has argued that graphical interfaces need to support mechanisms which will allow a smooth transition between different activities since their absence can lead to annoying errors. The criticisms made of Reichman's work in natural language processing do not apply to her work in window systems. Like Reichman, Cypher (1986) and Miyata

and Norman (1986) have also argued the need for supporting multiple activities within window systems. As far as the author is aware, her work has never been applied in a graphical environment before.

2.7 Multi-modal interfaces

As has already been discussed, natural language processing seems to have reached an impasse, at least for the moment, due to the sheer complexity of the endeavour. The continuing development of graphical interfaces reflects the fact that interacting with graphical objects, as in direct manipulation environments (Shneiderman, 1982, 1983), does, in many ways, circumvent the problems currently unresolved in natural language processing. However, as Hutchins et al. (1986) point out "when we give up the conversational metaphor, we also give up dealing in descriptions, and in some contexts, there is a great deal of power in descriptions." Human-human interaction, though, is not solely dependent on conversation. Other sensory functions are also brought into play by conversational participants, for example, sound, vision, touch, and gestures. To exclude their use from interface design is to impoverish the richness of the interaction, their inclusion by contrast will certainly complement the abilities of the user (Hutchins, 1987). This point has been noted by other researchers, Negroponte (1970), Baecker (1980a, 1980b), Buxton, Fiume, Hill, and Woo (1983), and Bolt (1981, 1984). The development of multi-modal interfaces, those which support video, speech, sound, combine natural language processing and direct manipulation techniques, are not new.

GRAFLOG

One approach which combines natural language and direct manipulation techniques to form a multi-modal system is described by Pineda (1988a), who states that a theory of interaction with computer graphical systems is needed

so that graphical representations can be correctly interpreted and that correct inferences can be drawn from them. In his system, GRAFLOG (Pineda, 1988a; Pineda, Klein, and Lee 1988b; Klein and Pineda 1990), a Computer Aided Design system, two representational systems, analogical and first-order logic (Prolog) are used to differentiate between different levels of abstraction, 'bottom' and 'top', of graphical symbols. Associations between these two different representational systems is achieved using linguistic interpretations through what he calls 'associative mechanisms'. The first mechanism, deictic expressions, are used to point out, in this case, graphical objects representing European cities. The second associative mechanism used is a developed graphical grammar which defines the semantic properties of graphical symbols and the relationships between them. Deictical references to graphical symbols are also defined in the grammar which specifies the relationships between symbols, that is, their geometrical properties (bottom level of abstraction) and their relationship to other symbols (top level of abstraction).

GRAFLOG is a system which uses natural language input to understand drawings, such as interpreting maps of Europe. In use, natural language statements are input in conjunction with deictic mouse actions to assign meanings to graphical symbols. For example, typing the statement "This is a city" (a deictical expression), followed by the selection of a graphical symbol (a deictical act), for example, a circle, will set set up the association between the geometrical class 'circle' and the property 'city'. Using an object-oriented style approach individual classes of objects may also be defined in a similar manner, for example, "This is Nice". Each input statement which is associated with a graphical object is encoded in a Prolog format, such as, City(Nice). In addition to handling associations, GRAFLOG can use Prolog to reason and make deductions about components that have been defined using a) properties of graphical objects typed in via deictic expressions, b)

logic rules associated with the objects and c) geometrical properties of the object itself.

GRAFLOG is one of a number of such attempts which seeks to exploit the linguistic channel of graphical environments in an effort to circumvent many of the unresolved problems of natural language. The approach is interesting and seems to have a great deal of potential, but as yet GRAFLOG is still only an experimental system. Other research which has sought to integrate natural language text and graphics includes the work of Klein (1987), Lee, Kemp and Manz (1989) on the ACORD project.

Other systems which exploit the combinational advantages of natural language and direct manipulation techniques are CUBRICON (Neal, Shapiro, 1988), XTRA (Wahlster, 1989), SHOPTALK and CHORIS (Cohen, Dalrymple, Moran, Pereira, Sullivan, Gargan, Schlossberg and Tyler 1989).

SHOPTALK and CHORIS

Extending the natural language question and answering system, Chat-80 (Warren, and Pereira ,1982), SHOPTALK and CHORIS allow graphical objects to be manipulated to resolve the difficult problem of anaphoric reference as found in everyday conversation. All three systems are currently being developed within the electronic manufacturing domain and employ a graphical front-end through which questions about the state of the factory can be posed. Analysis of user questions is performed by Chat-80 which transforms them into a subset of Prolog which is executed to provide an answer. Answers are then presented as tables, histograms or other display forms so that users may pose further questions if they wish. Questions posed in everyday conversational settings will almost certainly involve follow-up questions which will refer to previously mentioned entities. Reference to these entities is performed by the use of anaphoric reference using pronouns, for example, 'it' or deictical reference, for example, 'this'.

The resolution of referents for previously uttered noun phrases in natural language processing systems is very difficult. SHOPTALK and CHORIS attempt to resolve the problem of anaphora, the direct manipulation of context and deixis in a novel way.

The use of scrollable windows to delineate a particular context is used by SHOPTALK and CHORIS. Within these windows users can pose questions and receive answers. As well as responding to a question in natural language, the system displays entities within the answer as focus buttons which, when selected, can be used as a means of anaphoric reference. This has the effect of constraining the answer by indicating to the system the type of entity being focussed upon as well as its context delineated by the window boundary.

A current limitation of many natural language processing systems is that they are unable to construct a complete discourse structure tree during a user-system interaction. Instead, most form a restricted linear structure which is capable of handling limited anaphoric reference to entities previously mentioned. The need for such a structure is desirable (Grosz, 1981; Reichman, 1981) because it represents the history, past and present, of the discourse, in terms of context and focus. Tracking context switches, focus and anaphoric reference is facilitated by traversing the tree structure. SHOPTALK and CHORIS circumvent these difficult problems by constructing a graphical tree structure of queries through which users may navigate the discourse. Selecting a node, representing a previous query is equivalent to a 'conversational digression' and, selecting the most recent one a 'return move' á la Reichman.

Like GRAFLOG described above, SHOPTALK and CHORIS support deictical reference. GRAFLOG, as previously discussed, uses deictical action, performed first, followed by natural language input to define an association

between a symbol and its property. When SHOPTALK and CHORIS, by contrast, encounter a deictic reference in the natural language input its meaning is derived through a graphical deictic reference which is instantiated into the text string.

Another feature modelled by SHOPTALK and CHORIS is the support of natural language forms such as 'what', 'when' and 'to' which can be supplied with arguments, textual or graphic. This feature overcomes some of the limitations of direct manipulation interfaces in their inability to handle such forms and the relationships between them.

These two systems are impressive for they show that natural language and direct manipulation techniques can be used to supplement the other's deficiences. The weakness of this approach is that two different input mediums have to be used during the interaction, that of pointing with a mouse and the inputting of text. Ideally the interaction should be conducted with only *one* interactional style, namely direct manipulation. In addition, the interaction should be able to exploit the linguistic channel of graphical objects represented within the interface, as suggested by Draper (1986), so that phenomena like anaphora, ellipsis and deixis are supported.

Rather than using two types of interaction within an interface, Draper (1986) has suggested that the two types of interaction should be merged into a single form. He has identified many parallels that exist between conversation and interface design, for example, understanding the meaning of referring expressions within a particular context such as anaphora and ellipsis. Draper suggests that many features of conversation "should be meshed smoothly" with direct manipulation techniques. Supporting conversational mechanisms, for example, anaphora, ellipsis and deixis within a direct manipulation environment would enable users' activities to

be more effectively tracked, thus enhancing the smoothness of the interaction.

2.8 Summary

This chapter has provided a literature review of the major interactional styles which have been developed for interfaces. The aims of this review were to examine the various approaches to interface design with a view to establishing how they could be improved.

Command line dialogues are mainly used by expert users who wish to issue abbreviated commands to the computer and have them executed very quickly. Whilst command line dialogues continue to be popular they are not "user friendly" in that a) they make navigating the computer system very difficult for the novice user, b) the command line syntax is very intolerant of input errors and c) it takes time to learn the naming convention used by operating systems such as DOS, UNIX and CP/M.

The potential of natural language interfaces has much to offer the user, including freedom of expressiveness in the range of input accepted, no arcane input syntax to learn, and a 'natural' form of interaction. To achieve this though much more research needs to be carried out in the following areas: a) knowledge representation, b) the different types of mapping involved in understanding input sentences, and c) how the complexity of the mapping process increases as a sentence's constituent parts are changed. The three different methods of parsing, keyword matching, syntactic analysis and semantic grammars reviewed have all been used with varying degrees of success. The simplest, keyword matching, allows sentences which are ungrammatical or ambiguous to be recognized although it does not map the words onto structures that represent their meaning. The most complex, syntactic analysis, captures the richness of information contained

within a sentence by building up a detailed structure of its meaning using grammar rules. However, a sentence that is ambiguous will not be interpreted correctly and should it contain words unknown to the system the parsing process will fail. Semantic grammars represent the middle ground between these two approaches by parsing a sentence using concepts associated with a particular domain. This approach avoids many of the ambiguities that would arise during syntactic analysis. Since many syntactic generalizations are not supported within the grammar the number of production rules required to compensate for this may be large.

Menu-based systems have developed largely because of the computational complexity involved in constructing a natural language interface. As a communication medium they provide novice users with a short-hand way of expressing themselves without having to learn a complicated input syntax. On systems which support deep menu structures navigating a range of menus can be cumbersome and displaying them can take up a large amount of screen space. Despite these criticisms menu-based systems are extremely popular and widely used in many popular computing packages.

Direct Manipulation interfaces allow icons representing objects or concepts to be manipulated giving users immediate visual feedback of their actions and the results of these actions. The feeling of directness experienced by a user depends upon, not least, a) the minimization of task-mapping required to understand the meaning of icons representing domain objects and concepts, b) how closely domain objects mimic the behaviour of real objects, and c) how much control they feel they have over these objects. Despite the widespread use of direct manipulation interfaces and the advantages they seem to offer to users, they are not well understood. Much more research on the links which exist between the semantics of the domain and the semantics of the interface need to be carried out.

The development of window systems has opened up the possibility of enhancing the interaction process a) by acting as a communication medium within which all user input is typed, b) by contextualising a user's activities and c) by acting as an extension to a user's own internal memory by displaying and keeping track of information. Current window systems are limited because they do not show the relationship which may exist between two or more windows. Since users structure their tasks to carry out multiple activities, each of which compete for attention, then the design of window systems should support and facilitate these activities. In particular, visual and contextual cues should be provided to support the suspension and resumption of interrupted contexts.

Multimodal interfaces attempt to exploit the potential of human-sensory apparatus, including, sound, vision, speech and gestures, all of which can be used to convey information to the computer. These types of interfaces are still in the embryonic stages of development, but it is too early to say how successful they will be, although they do hold a great deal of promise. However, multimodal interfaces which combine natural language and direct manipulation techniques have been around for quite some time. These two interaction styles, when used together, overcome the limitations of the other when used seperately, as in the case when deictical pointing actions are used to resolve natural language anaphoric references. A major drawback of this approach is that two different interactional styles have to be used, which, it can be argued, places an extra cognitive burden upon the user.

2.9 Conclusions

The conclusions to be drawn from the review are that no one interactional style can claim superiority over another because each has merit and its choice will largely be determined by the type of environment it has to operate

in (for example, electronics, education). The use of natural language with its historical precedence has a clear influence on interface design which is reflected, not only in natural language processing systems, but also within graphical interfaces. Overcoming the limitations of natural language input with that of direct manipulation techniques does provide users with a greater degree of expressiveness when using one of these interactional styles individually. However, two separate interactional styles have to be used which may reduce the feeling of 'direct engagement' experienced by a user.

The idea of merging natural language mechanisms with direct manipulation techniques within one interactional style, as proposed by Draper (1986), seems to be a far better approach than having to continually move between two different input mediums. It is proposed that the models of interaction developed within this thesis should take account of Draper's work.

The review in this chapter has also shown that users do not engage in single isolated activities, but rather in multiple activities between which they alternate. Since the research work here aims to construct an interactional model that takes into account how users work, the approach advocated by Cypher, Miyata and Norman, and Reichman will be pursued.

A model of interaction which is capable of exploiting the linguistic channel of a graphical interface, supporting phenomena like anaphora, ellipsis, deixis and transitions between related activities, implies a phased approach to its development.

In the next chapter a prototype model, Circuit I, is presented which shows how mouse selections corresponding to command and argument variations can perform analogous functions to natural language anaphora, ellipsis and deixis.

Circuit I : Modeling SOPHIE dialogue

3.1 Introduction

This chapter describes an object-oriented prototype, Circuit I (Singer, 1990), which has been constructed to assess the dialogue capability of SOPHIE in a contemporary context. That is, to establish the feasibility of modelling the natural language handled by SOPHIE graphically. In particular, surface linguistic features supported by SOPHIE are modelled to determine if graphical interfaces can provide some of the functionality of anaphora and ellipsis as used in natural language.

An overview of the internal and interface design of Circuit I is given which describes the different objects and the mechanisms which control them. Examples showing how objects representing circuit components and faults may be combined by users to pose simple and complex questions, are also given.

Given that graphical treatment of surface linguistic phenomena is possible, a means by which graphical interaction may be extended to support conversational digressions is discussed. The chapter then concludes with a summary.

3.2 Circuit I - internal design

To assess the dialogue capability of SOPHIE in a contemporary context, Circuit I, an object-oriented prototype, was constructed so that a small sample of SOPHIE dialogue, given below, could be modelled graphically.

- SOPHIE dialogue

Requesting measurements

What are the specifications of Q4?
What is the resistance of R22?
What is the voltage between node 4 and node 5?

Modifying the instrument

Short R3
Open R1

Ellipsis

What is the voltage across R1?
Across R2?
R3?

Anaphoric-like statements

What is the voltage across R2?
Its current?
Resistance?

Written in HyperCard, Circuit I is a simple series-parallel circuit (see figure 3.2) with which users interact using direct manipulation and menu-driven techniques (see figure 3.1). No computational model is used to calculate values for the circuit, instead it references look-up tables which contain values which are used to answer questions posed to it graphically.

3.2.1 Objects

Circuit I's interface comprises twenty-nine objects which represent a) the components which make up the series parallel circuit, b) the multimeter buttons and its meter display, c) the reset button, d) the circuit dialogue box, e) the fault insertion box, f) the circuit card, g) the circuit background card, h) the circuit background voltage, current and resistance tables and i) the circuit stack. In addition, two pull-down menus are used to derive

information or effect a change to these objects. With the exception of the menus, each of these objects are associated with a 'script' which describes the behaviour of that object when activated.

3.2.2 Hierarchy of objects

The objects used by Circuit I are hierarchically structured to form a 'stack'. The objects used can be categorized by their type, each of which has its own place within the hierarchy. Objects representing 'buttons' (circuit components, multimeter, reset) and 'fields' (meter display, circuit dialogue, circuit faults) occupy level 1, the highest level in the stack. At level 2 is the circuit card through which all messages from level 1 objects pass. Immediately below this at level 3 is the circuit background card which has three fields associated with it containing data used by the system. Level 4 is the current stack representing Circuit I to which all preceding levels are linked. The remaining levels, 5 and 6, are associated with HyperCard's 'home stack' and the application program HyperCard itself which will not be discussed here.

When Circuit I is activated by double-clicking on the icon representing its stack, first, the HyperCard application is loaded followed by the 'home stack'. Next, Circuit I is loaded which resides in the computer's working memory as illustrated in figure 3.1 below.

3.2.3 Scripts and handlers

Circuit I uses seventeen scripts which tell the objects they control how to behave under different circumstances. Each script used contains one or more mouse 'handlers' which when taken together describe the behaviour of Circuit I's stack in response to 'messages'. Of the seventeen

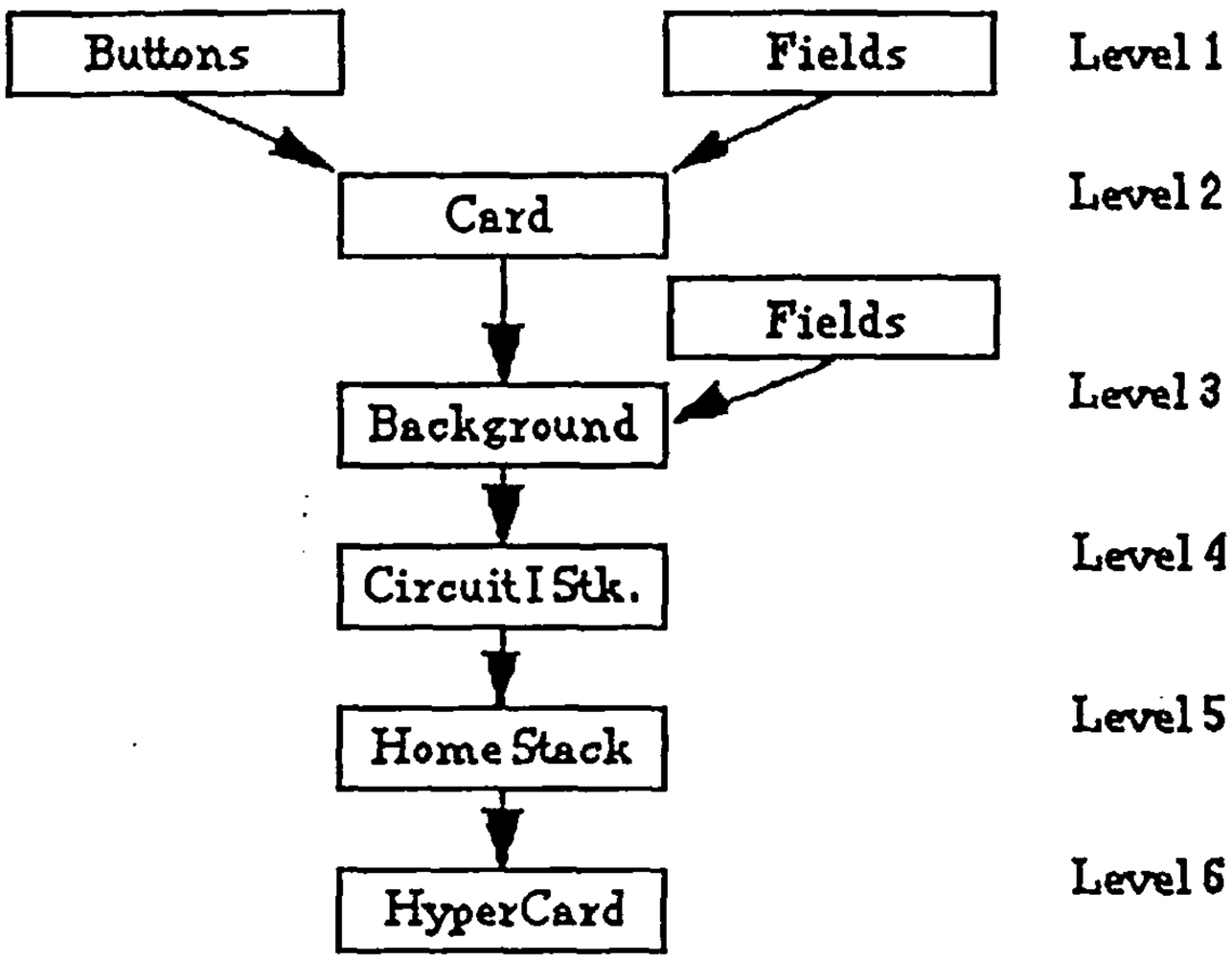


Figure 3.1 The object hierarchy of Circuit I

scripts used by Circuit I: At level 1, eleven are used for circuit components representing objects, i.e., resistors R1, R2, R3 and a bulb; inspection points P1 through P7. Four are for buttons representing actions of the multimeter which takes voltage, current and resistance measurements, and a reset object which de-highlights all activated buttons. Level 2 uses one script for the circuit card which contains mouse handlers to control activation of the 'specification' and 'fault' pull-down menus. When, for example, a component is faulted using the fault menu, the mouse handlers in this script update the tables containing the voltage, current and resistance data stored at level 3. The circuit background card at level 3 has no handlers associated with it but does have three background fields. These fields, as previously described, hold data for the voltage, current and resistance tables. Each field has a script associated with it which retrieves data from the table to answer user-posed questions when directed to do so. The stack of Circuit I at level 4 is used to initialize all objects and variables used in the interface and to control HyperCard menus which are displayed.

3.2.4 Messages

The activation of one object has the effect of passing a message to other objects which, if activated also, informs them what appropriate action(s) to take. There are two main sources of messages in Circuit I, those from the mouse and those from menus which are used to signal some required command-argument action. To illustrate this, consider the case of taking a voltage measurement across a resistor, for example, R1. When the object R1, (representing an argument), is activated by the mouse it changes colour from white to black and sends a message to the voltage button (representing a command). If the voltage button has been previously activated then this command-argument combination, identified by id (identification) numbers, is used to retrieve the correct voltage reading across R1 from the voltage table of values. In the reverse case, where the object R1 has been activated first, the system will wait until the voltage button has been selected before retrieving data from the voltage table of values. As well as supporting these types of combinations the command or argument portions of the syntax can also be varied as will be shown in §3.3.

3.2.5 Summary

The internal design of Circuit I is a hierarchy of objects whose relationships with one another are determined through messages which are interpreted via mouse handlers within an object's script. Using simple combinations of objects representing commands and arguments which may be varied, the system demonstrates how mechanisms which perform analogous functions to those like anaphora and ellipsis in natural language may be supported. It is interesting to note that, the command-argument/argument-command syntax is roughly analogous to the "verb-noun/noun-verb" form of syntax although from a linguistic perspective this would not be technically correct.

The current implementation of Circuit I supports menu-style interaction for

faulting components so as to constrain the range of ways that they can be modified. Circuit I's lack of a proper computational model severely restricts the types of modifications that can be made, a problem which needs to be rectified in any future implementation.

3.3 Circuit I - interface design

This section describes the different parts of Circuit I's interface and their functions. It also discusses the questions which can be expressed, in terms of the combined selections of these parts. Circuit I's graphical interface is made up of a number of different parts each of which fulfill a specific function. These parts are described below.

3.3.1 Multimeter

Three buttons (Ohms , Amps , Volts) are used which when individually depressed are highlighted to indicate the current type of measurement being asked for. Collectively, these buttons act like a multimeter which can be used to measure these different concepts.

3.3.2 Components

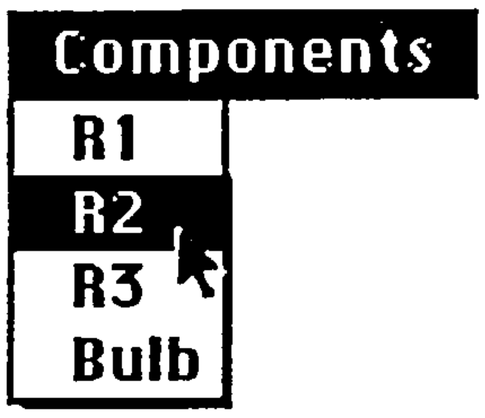
Circuit I consists of eleven buttons, i.e., seven inspection points (P1 - P7), four components (R1 - R3) and a bulb which are connected together to form a series-parallel circuit.

3.3.3 Pull-down menus

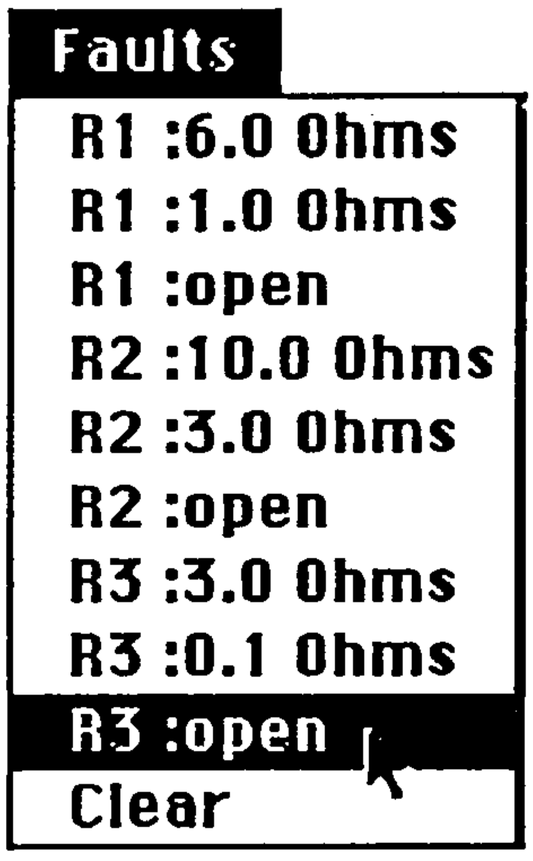
In addition to these buttons two pull-down menus are used to (a) request a component's specification and (b) to modify a component's value. These menus are illustrated below. Normal and faulted circuit values share the same three arrays (Current, Voltage and Resistance), normal circuit values being restored to these arrays by selecting the 'Clear' command from the

'Faults' pull-down menu.

Specification menu



Fault menu



3.3.4 Posing graphical questions

Circuit I (see figure 3.2) shows how questions handled by SOPHIE may be expressed in a graphical fashion.

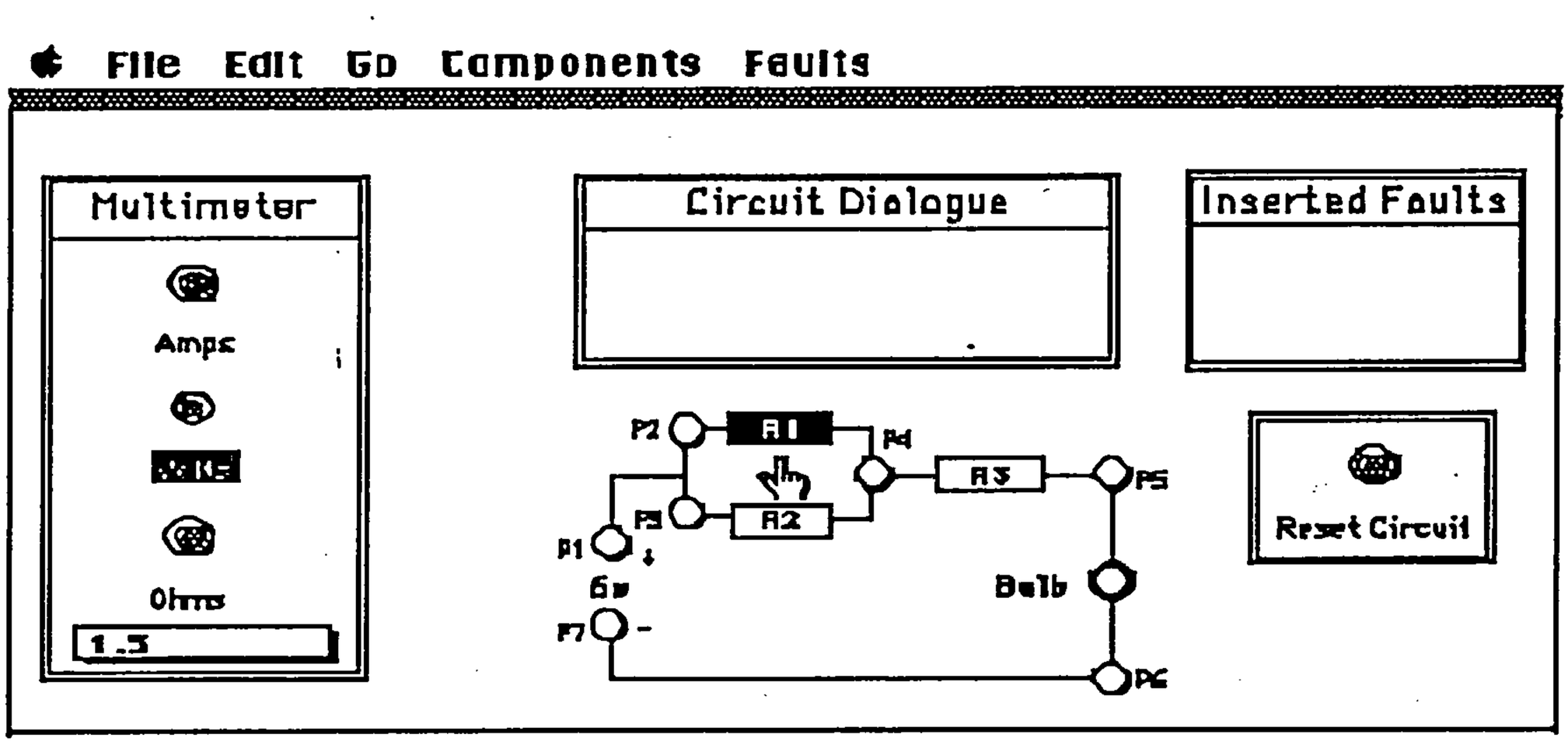
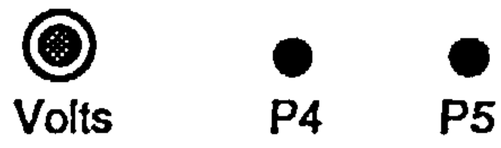


Figure 3.2 A command-argument combination: "What is the voltage across R1?"

The following examples below show SOPHIE dialogue on the left-hand side of the screen and its equivalent expressed graphically on the right.

1. SOPHIE Dialogue **Graphical Dialogue**



>> WHAT IS THE VOLTAGE BETWEEN NODE 4 AND NODE 5?

User Actions

a. To highlight the 'volts' button the user would depress the mouse pointer over it. That is:

(i) MouseDown

Effect: highlight the Volts button

(ii) MouseUp

Actions (i) and (ii) (in this context) correspond to the natural language utterance "**What is the voltage**"

b. To highlight the inspection point P4 the user would depress the mouse pointer over it. That is:

(iii) MouseDown

Effect: highlight the inspection point P4

(iv) MouseUp

Actions (iii) and (iv) corresponds to the natural language utterance " **between P4**".

c. To highlight the inspection point P5 the user would depress the mouse pointer over it. That is:

(v) MouseDown

Effect: highlight the inspection P5

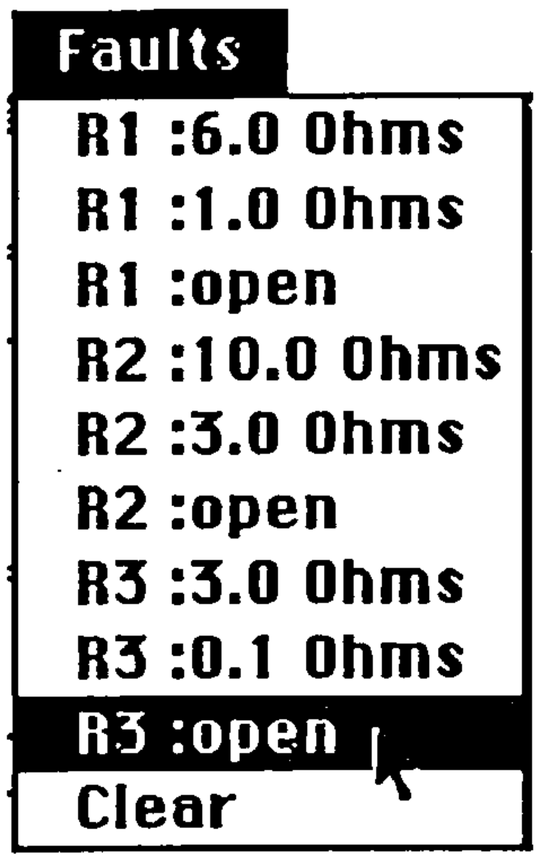
(vi) MouseUp

Effect: display the voltage reading

Action (v) corresponds to the natural language utterance "**and P5**" whilst (vi) executes the command and displays the voltage between P4 and P5 in the

dialogue box.

2. SOPHIE Dialogue **Graphical Dialogue**



>>OPEN R3

User Actions

a. To make resistor R3 'open' the user would depress the mouse pointer over the 'Faults' pull-down menu and highlight the option 'R3 :open'. That is:

(i) MouseDown

Effect: open Faults pull-down menu

(ii) DragMouse

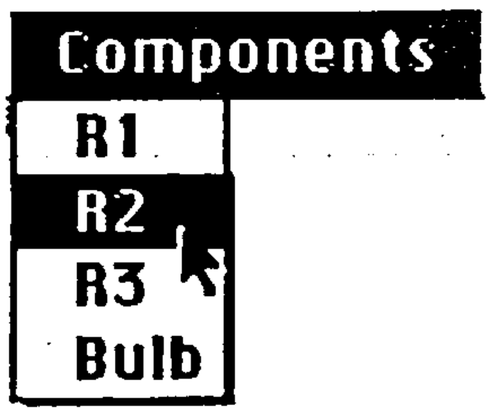
Effect: highlight 'R3 :open' option

(iii) MouseUp

Effect: makes resistor R3 open circuit.

Action (i) selects the range of permissible faults (ii) corresponds to the natural language utterance "**Open R3**", whilst action (iii) executes the command.

3. SOPHIE Dialogue **Graphical Dialogue**



>>WHAT IS THE SPECIFICATION OF R2?

User Actions

a. To obtain resistor R2's specification the user would depress the mouse pointer over the 'Components' pull-down menu and highlight the option R2. That is:

(i) MouseDown

Effect: open Components pull-down menu

(ii) DragMouse

Effect: highlight the option 'R2' option

(iii) MouseUp

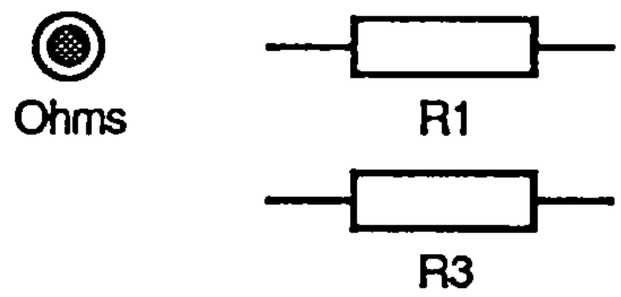
Effect: display resistor R2's specification

Action (i) corresponds to the natural language utterance "**What's the specification of**", (ii) corresponds to the utterance "**R2**" whilst (iii) executes the command and displays the specification of R2 in the dialogue box.

4. SOPHIE Dialogue **Graphical Dialogue**

>>WHAT IS THE RESISTANCE OF R1?

>>R3?



User Actions

a. To highlight the 'ohms' button the user would depress the mouse pointer over it. That is:

(i) MouseDown

Effect: highlight the Ohms button

(ii) MouseUp

Actions (i) and (ii) correspond to the natural language utterance "**What is the resistance of**"

b. To highlight the resistor R1 the user would depress the mouse pointer over it. That is:

(iii) MouseDown

Effect: highlight the resistor R1 button

(iv) MouseUp

Action (iii) corresponds to the natural language utterance "**R1?**" whilst action (iv) executes the command.

c. To highlight the resistor R3 the user would depress the mouse pointer over it. That is:

(v) MouseDown

Effect: de-highlights resistor R1 button

highlights the resistor R3 button

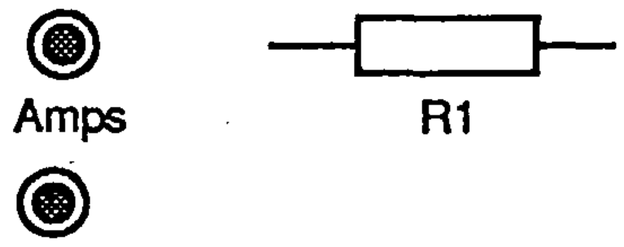
(vi) MouseUp

Actions (v) corresponds to the natural language utterance "**R3**" and constitutes a graphical ellipsis. Action (vi) executes the command.

5. SOPHIE Dialogue **Graphical Dialogue**

>>WHAT IS THE CURRENT THRU R1?

>>WHAT IS THE VOLTAGE ACROSS IT?



User Actions

a. To highlight the 'amps' button the user would depress the mouse pointer over it. That is:

(i) MouseDown

Effect: highlight the Amps button

(ii) MouseUp

Actions (i) and (ii) correspond to the natural language utterance "**What is the current through**"

b. To highlight the resistor R1 the user would depress the mouse pointer

over it. That is:

(iii) MouseDown

Effect: highlight the resistor R1 button

(iv) MouseUp

Action (iii) corresponds to the natural language utterance "**R1?**" whilst (iv) executes the command and displays the current through R1 in the dialogue box.

c. To highlight the 'volts' button the user would depress the mouse pointer over it. That is:

(v) MouseDown

Effect: highlight the Volts button

(vi) MouseUp

Action (v) corresponds to the natural language utterance "**What is the voltage across it?**", constituting an anaphoric-like statement, whilst (vi) executes the command and displays the voltage across R1 in the dialogue box.

3.4 Using Circuit I

In use, Circuit I emulates all of the dialogue described in §3.2, in the manner described above in §3.3. A vital part of SOPHIE's attractiveness to users was its ability to handle anaphora and ellipsis. The sample of SOPHIE dialogue modelled is sufficient to demonstrate a graphical direct manipulation equivalent of these features.

3.4.1 Command-argument combinations

SOPHIE allowed its users to make measurement requests through its natural language interface by posing questions such as, "What is the voltage

across R1?". Circuit I supports such "What..." requests by allowing users to select objects corresponding to commands and arguments as described in §3.2.4. To pose this SOPHIE question to Circuit I the user would select the command "measure voltage" with the argument "R1" as shown in figure 3.2 to obtain an answer.

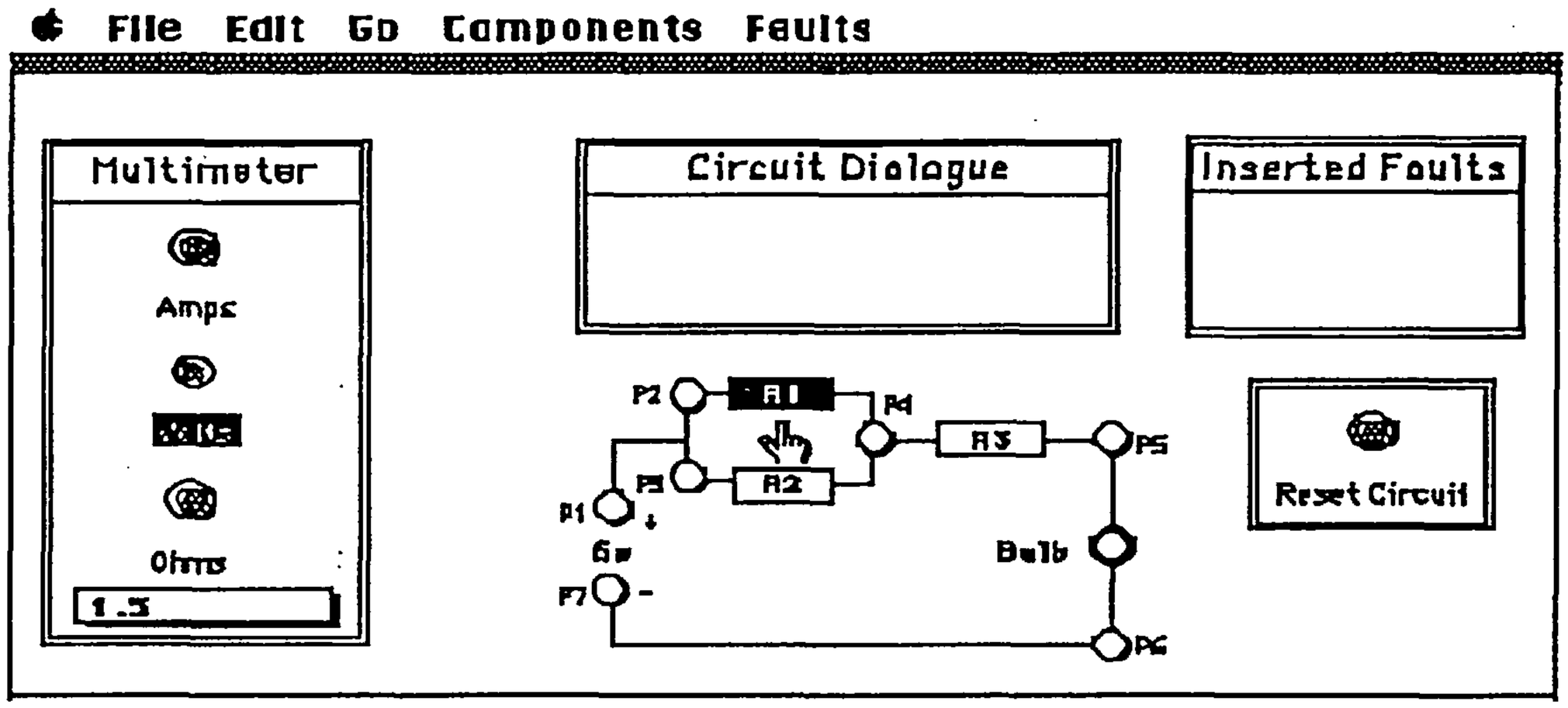


Figure 3.2 A command-argument combination: "What is the voltage across R1?"

SOPHIE also handled more complex measurement requests of the form "If x then y", for example, "What is the voltage across R2 if its resistance is 3 ohms?". These hypothetical types of "What...if..." questions can be posed to Circuit I, albeit crudely, by extending the command-argument syntax to support the selection of another command. In this case the SOPHIE question would be posed to Circuit I by first selecting the command "measure voltage", then the argument "R2" and finally by selecting the command "make R2 3ohms" via the Faults pull down menu as shown in Figure 3.3.

3.4.2 Varying arguments

The use of semantic grammars employed by SOPHIE allows it to recognize elliptic utterances, that is utterances that do not express complete thoughts, a complete question or command. Instead, only differences between the

intended thought and an earlier one are given (Burton et al., 1979).

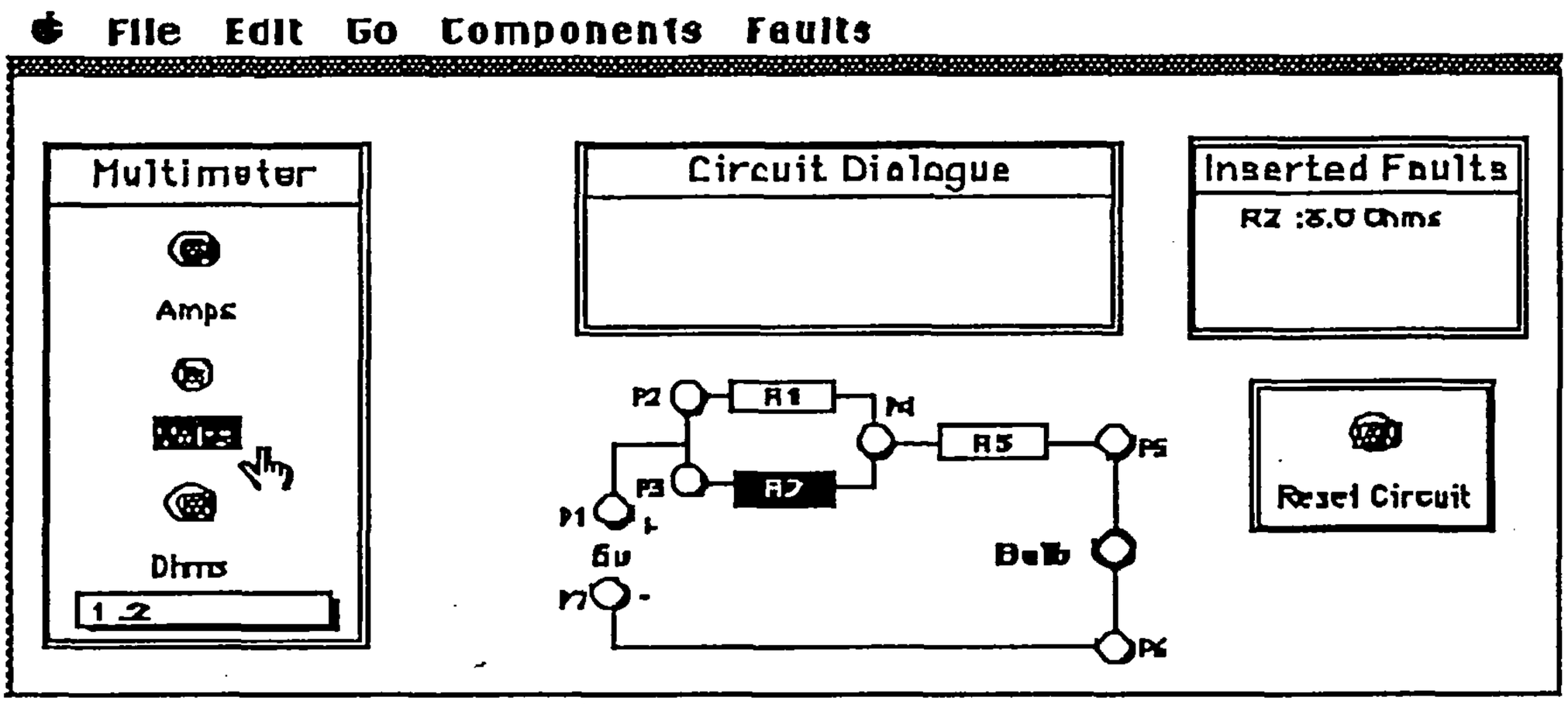


Figure 3.3 Command-argument-command combination: "What is the voltage across R2 if its resistance is 3 ohms?"

For example, SOPHIE handled elliptic utterances likes those in statements 2 and 3 of excerpt 1.

Excerpt 1

1. What's the resistance between P2 and P4?
2. P4 and P5?
3. P5 P6?

Mouse selections which perform the function of ellipsis are supported in Circuit I by varying the argument portion of the syntax. For example, statement 1 of excerpt 1 can be posed to Circuit I by selecting "measure resistance" with the arguments "P2" and "P4". The ellipsis which occur in statements 2 and 3 can be performed by selecting the argument “P5”, deselecting “P2” and “P6” deselecting “P4” as illustrated in figures 3.4 to 3.6.

3.4.3 Varying commands

SOPHIE could also recognize anaphoric references which occurred in its natural language input. For example SOPHIE resolves anaphors like those

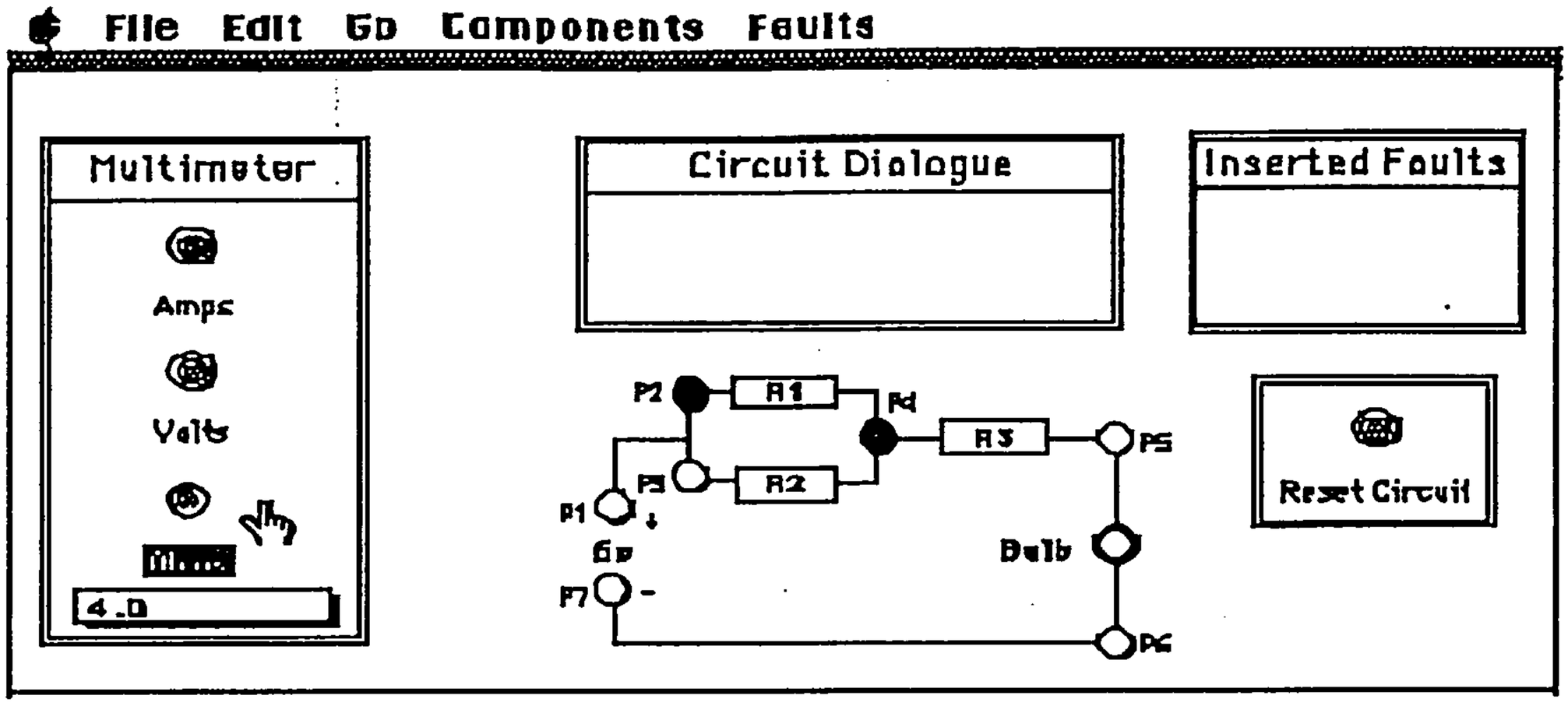


Figure 3.4 Command-argument-command combination: "What is the resistance between P2 and P4?".

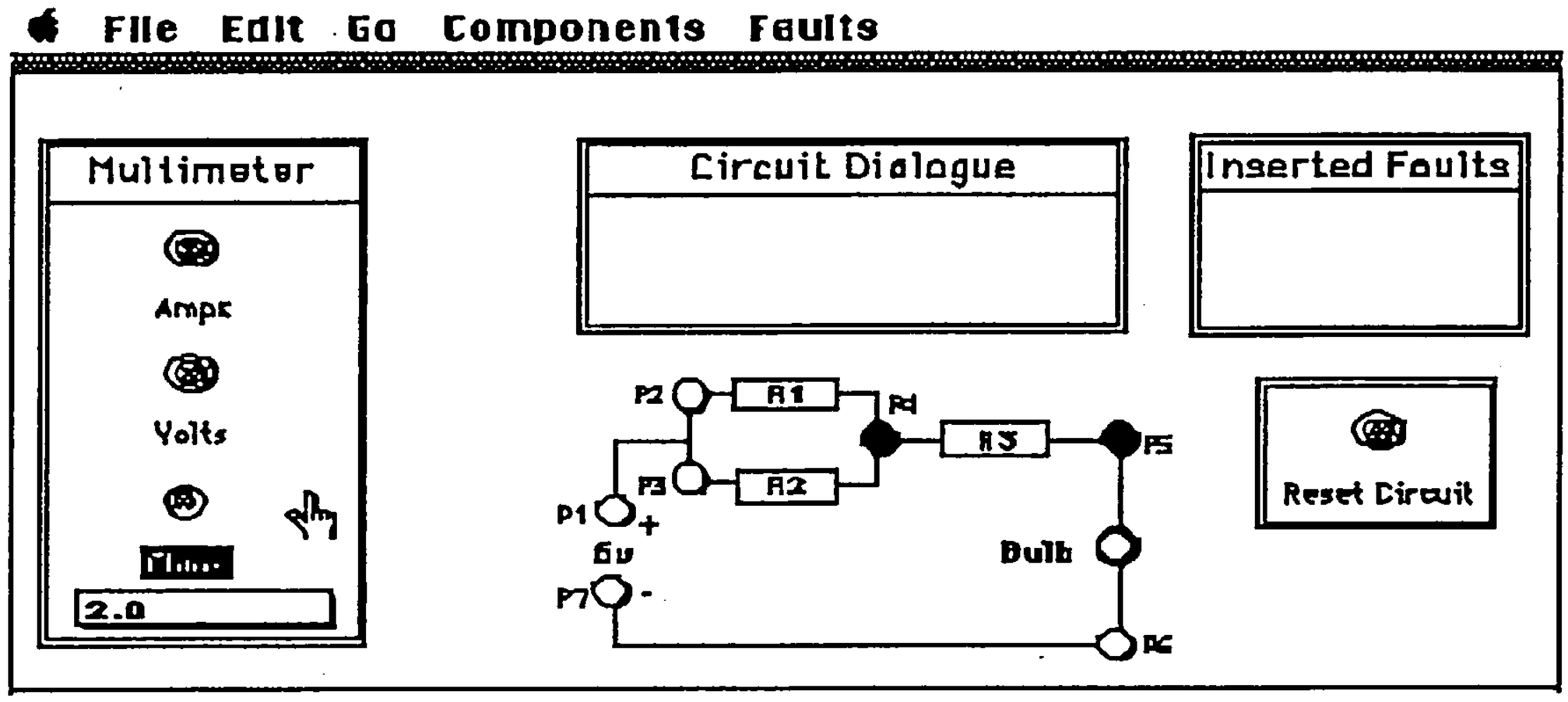


Figure 3.5 Varying the argument : P2 in favour of P5.

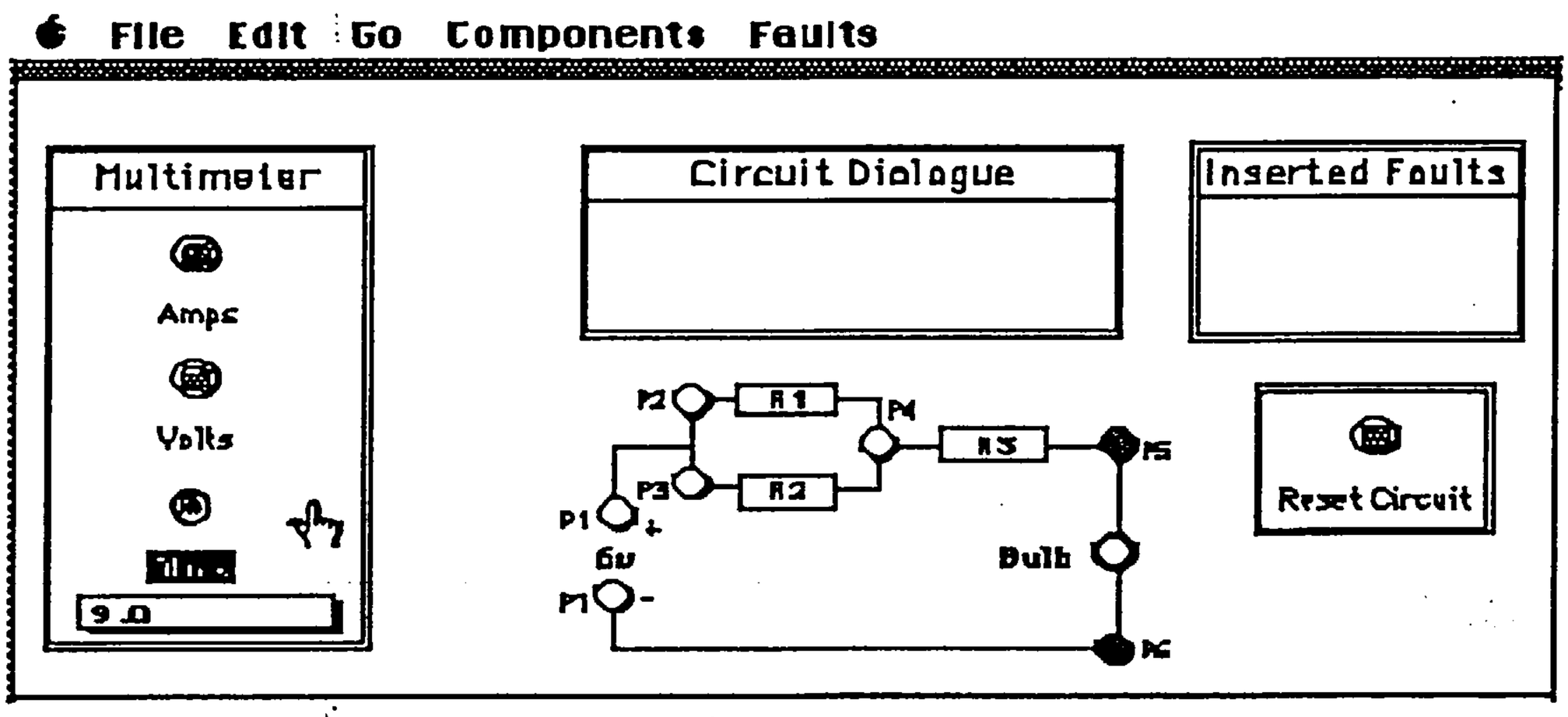


Figure 3.6 Varying the argument : P4 in favour of P6

occurring in statements 5 and 6 of excerpt 2.

Excerpt 2

4. What is the voltage across R1?
5. What about **its** resistance?
6. And **its** current?

In statements 5 and 6 "its" is an anaphor looking for a pronominal referent which in this case is the argument "R1". Circuit I's syntax also allows users' mouse selections to perform a similar function to answer such types of questions by allowing the user to vary the command portion of the syntax. To illustrate, the command-argument combination representing "measure voltage" and "R1" is made and then, by varying the commands "measure resistance" and "measure current" in statements 5 and 6 the anaphors can be resolved. Figures 3.7 to 3.9 illustrate how statements 4, 5 and 6 are handled by Circuit I.

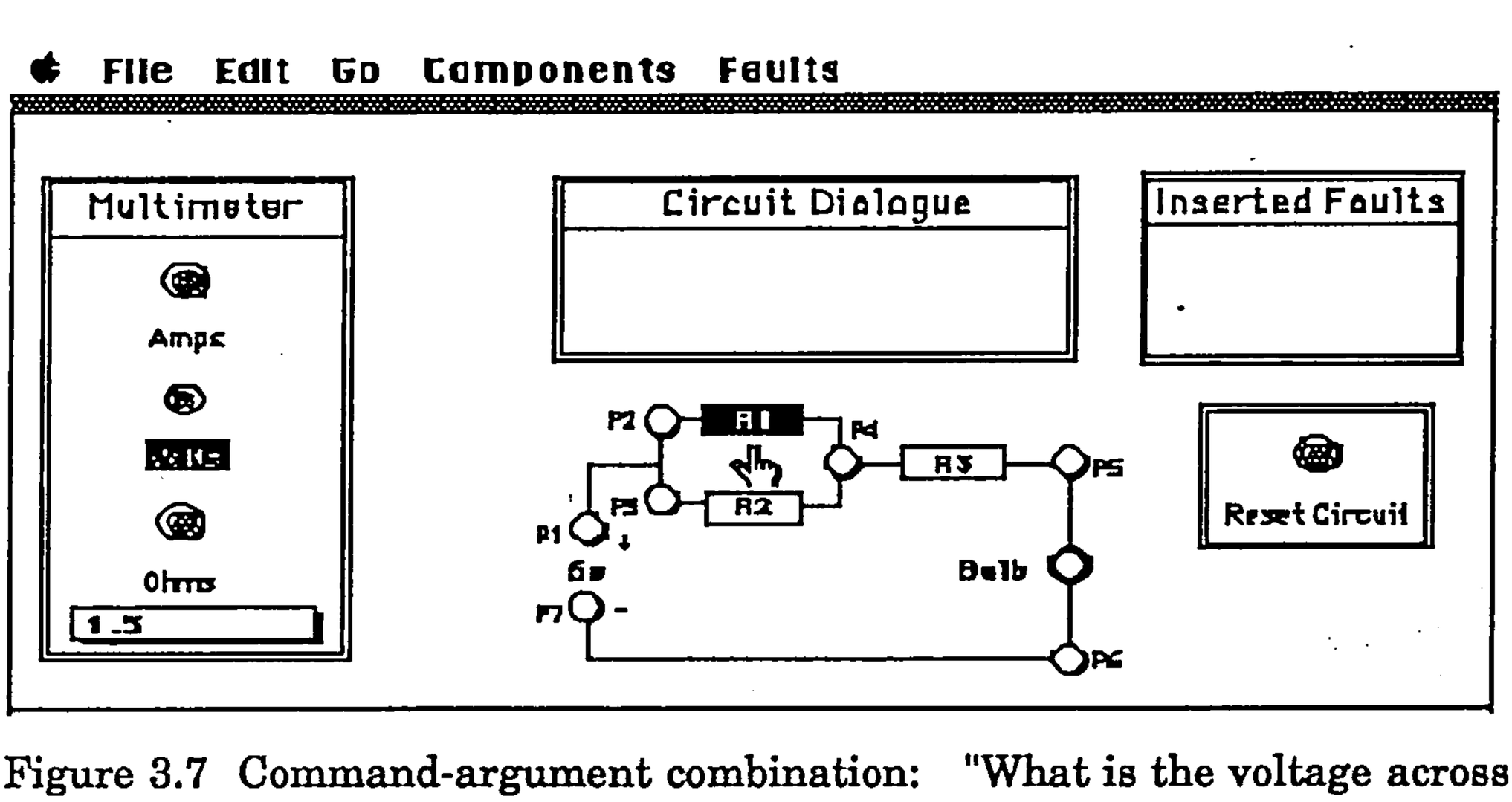


Figure 3.7 Command-argument combination: "What is the voltage across R1?"

3.4.4 Summary

The graphical mechanisms supported in Circuit I, through their use of pull-down menus and direct manipulation techniques, allow users to express a

range of simple SOPHIE dialogue. Simple questions posed to Circuit I

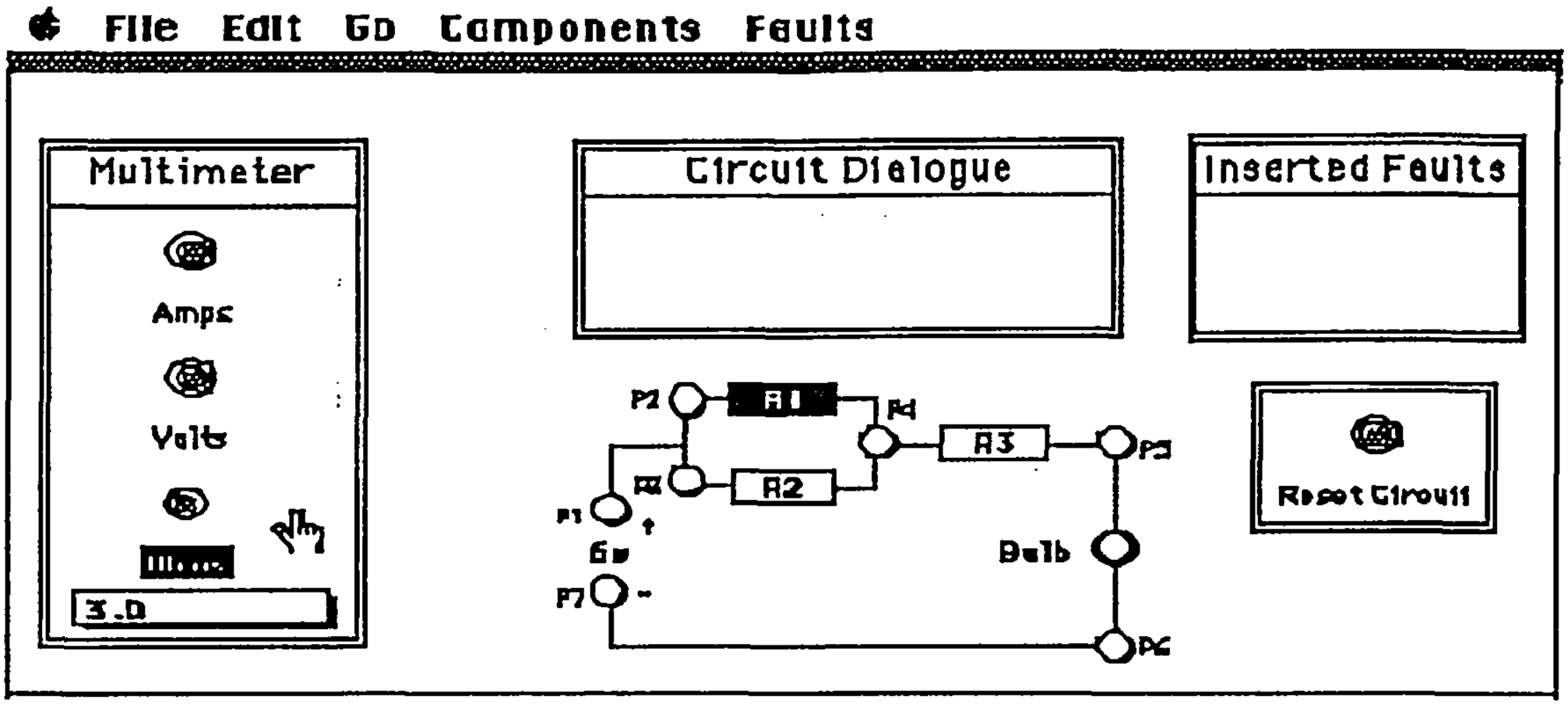


Figure 3.8 Varying the command portion: "What about its resistance?".

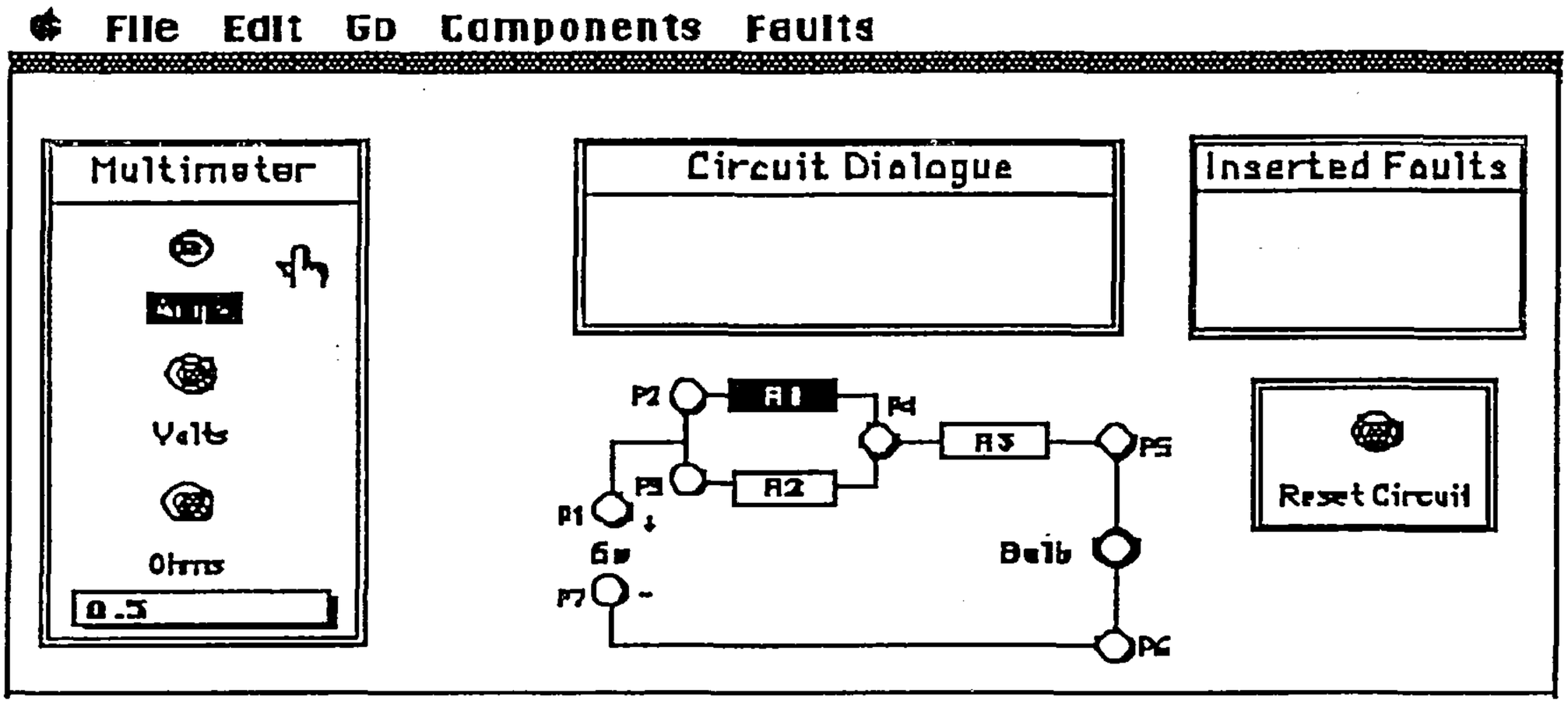


Figure 3.9 Varying the command portion: "And its current?".

consists of two parts, a command supplied with an argument. Changing the argument portion of a question performs an analogous role to that of ellipsis expressed in natural language.

Similarly, altering the command portion in favour of another performs an analogous role to that of anaphora.

Supporting this type of *free-order* syntax is a key feature of Circuit I because it extends the way in which a user may express themselves within the interface. In particular, the use of free-order syntax promotes the

expression of ellipsis but without the constraints imposed upon its natural language counterpart. Natural language ellipsis, in its strictest sense, requires that the ellipted part of a sentence should be unambiguously specifiable (Crystal, 1980). For example, consider the following SOPHIE dialogue:

Excerpt 3

7. What is the voltage at Node 5 if the load is 100?
8. 6?

As Burton et al. (1979) point out, interpreting statement 8 of excerpt 3 causes ambiguity, SOPHIE favouring its interpretation "as a load rather than a node because that was the last number mentioned." Within spoken dialogue an elliptical expression, as in statement 8, would probably not be used. Because Circuit I's free-order syntax avoids such ambiguities it promotes the expression of ellipsis, in fact probably more than in spoken or written natural language. Avoiding such ambiguities allows Circuit I to cover all the cases of ellipsis which occured in SOPHIE which would have to be done by some mixture of both ellipsis and anaphora.

Complex 'What.. If..' questions may also be posed to Circuit I which are composed of two arguments and two commands, whose parts and faults may also be changed. A limitation of this approach at present is that commands representing faults can only be changed by using pull-down menus. Commands representing measurements and arguments representing components should be changed using direct manipulation techniques. This deficiency will be rectified in future developments of this model.

3.5 Extending Circuit I

The analysis of Circuit I has demonstrated not only that anaphora and ellipsis have graphical counterparts, but that the support of such discourse phenomena may be used to facilitate the expression of subjects' intentions through their mouse-clicking actions. Circuit I is an extremely limited example, but it is suggested that supporting such phenomena allows it to 'remember' the surface and deep structure of users' intentions within a context. In the current implementation the two contexts supported, normal and faulted circuits, are treated as separate and unrelated. Moving between these two contexts is effected through the use of a pull-down menu, either by inserting a fault or by removing it. When a transition from a normal to a faulted circuit context is made the state of the previous context is not remembered so that when it is returned to it has to be re-instated. In deciding how to extend the current implementation, consideration was given as to how subjects' intentions could be tracked between contexts using conversational principles. This section describes one possible approach to achieve this.

3.5.1 Discourse model

In addition to tracking users' intentions within a context, Circuit I will also need to be able to construct a discourse model which allows it to partition a conversation into "context spaces" (Reichman, 1978a). By analysing the conversational flow in person-to-person communication to establish what makes a conversation coherent, Reichman has tried to spell out people's discourse rules explicitly. She has identified "the context space as the fundamental unit of discourse processing. The structure of a discourse can be specified by the identification of its context spaces and the relations between them." (Reichman, 1985).

3.5.2 Context spaces

Although Circuit I does not construct a discourse model as it tracks users' intentions at the moment, it might be possible to do so by providing it with the means to partition the objects clicked on by the mouse into a related hierarchy of graphical context spaces. Such context spaces could provide Circuit I with the means by which to engage in an extended graphical dialogue with users.

Context spaces could be modelled within a window environment, where the boundary of the window corresponds to the context space boundary. Two windows could be used to support the normal and faulted contexts of Circuit I, each being activated by a switching mechanism to effect a digression from one context to another. Moving from a normal working context to a faulted one would have the effect of saving and suspending the normal context during the digression. Returning to a suspended context would re-instate it again allowing the user to continue where they left off before the digression. Ideally, the mechanism should allow users to move smoothly between these two contexts. One possible mechanism is that of 'cue' phrases suggested by Reichman (1986).

3.5.3 Cue phrases

Cue phrases, as used by speakers during a conversation, are surface linguistic signals which indicate to listeners that a change of topic is about to occur. Reichman (1985) has identified a number of these cue phrases which "signal that a context space boundary point has been reached; and simultaneously they specify the kind of shift (the kind of conversational move) about to take place."

Reichman specifies two types of cue phrases which are used to signal the 'interruption' of a currently active context space, and a 'return to a

previously' interrupted context space. By modelling these cue phrases in Circuit I, "Incidentally...; By the way..." and "Anyway...; In any case", it is suggested, could provide a mechanism which would allow smooth transitions to occur between normal and faulted contexts.

3.6 Summary

The work presented in this chapter has provided an overview of Circuit I, a prototype graphical interface capable of supporting a limited sample of SOPHIE dialogue. The internal design of Circuit I uses HyperCard's object-oriented environment to model objects, command and arguments. Messages can be passed between these objects representing actions to be performed upon objects or information to be displayed to users.

Circuit I's interface contains objects with specific visual properties, i.e., components representing resistors, a bulb, a battery, a multimeter and menu bars for applying faults and obtaining component specifications. As discussed, each object represents a command or argument, and their combined activation via mouse actions allows simple and complex questions to be posed.

Varying the different objects representing commands and arguments allows users' mouse selections to perform analogous functions to anaphora and ellipsis which SOPHIE supported. This, it is suggested, provides users with a more economical way of carrying out mouse selections than is the case in those interfaces not supporting such mechanisms. The support of free-order syntax within Circuit I promotes the expression of a more general form of ellipsis by overcoming the constraints inherent in its natural language counterpart. This allows, in interactive sequences, Circuit I to approach the brevity, naturalness and convenience of expression that SOPHIE achieved through its use of natural language anaphora and ellipsis.

Circuit I is a limited example of how objects representing components in a graphical interface can use linguistic analogy to model SOPHIE dialogue. Reichman's concept of windows as context spaces and cue phrases to effect a conversational digression between related contexts offers one way in which the work presented here could be extended.

The next chapter describes an empirical study conducted with Circuit I.

Summative Evaluation of Circuit I

4.1 Introduction

This chapter gives details of an empirical study conducted with Circuit I over a 10 day period. Details of the study are described, including the subjects used, the aims of the study, the methodology adopted and the results obtained.

4.2 Aims of Study

The aim of the study was to use Circuit I to establish whether a) users associated natural language dialogue with graphical actions, and b) certain graphical actions performed an analogous role to natural language anaphora, ellipsis and deixis.

4.3 Subjects Used

Four subjects, all volunteers from the Electronics Faculty at Walton Hall, Milton Keynes, took part in the study. Each subject was a qualified electronics troubleshooter, with a minimum of ten years experience. None of the subjects had seen or used the system prior to the study.

4.4 Methodology

Two main approaches were used to capture data from subjects during this study, that is, *direct observation* and *verbal protocols*.

Given the unfamiliarity of subjects with Circuit I and how it should be used, directly observing a subject has the benefit that, if they encounter difficulties corrective assistance can be given immediately. The disadvantage of this approach is that it can alter a subject's performance level, via the

Hawthorne effect (Algers, et al., 1990), as the subject is constantly aware that their performance level is being monitored.

The other technique used during the observational study was to record subjects' concurrent think-aloud protocols on a standard tape recorder. Subjects were free to mention anything that came to mind as they performed the set tasks. The aim of recording subjects' protocols was to analyze them against subjects' mouse-clicking actions to establish the degree of correspondency between the two. It has been argued that concurrent think-aloud protocols (Newell & Simon, 1972; Ericsson and Simon, 1980) represent information currently being attended to in a student's short term memory and hence indirectly represent the cognitive processes that are being brought to bear to solve a problem or understand a task. Whilst this position has been challenged by others (Nisbett and Wilson, 1977), Ericsson et al., (1980) found that the conditions under which Nisbett and Wilson considered verbal reports to be valid were consistent with their own findings. In the context of problem solving "...verbal reports, elicited with care and interpreted with full understanding of the circumstances under which they were obtained, are a valuable and throughly reliable source of information about cognitive processes." (Ericsson et al., 1980)

No time restriction was placed on subjects to perform tasks given. All were given the same set of tasks to do, as shown in Table 4.1, and once completed subjects were free to explore the environment further or ask any questions they wished.

By adopting a participative approach, it was hoped that subjects would be more willing to explain the motivations behind their actions, thus giving a deeper insight into their reasoning processes.

Tasks
1. Determine the resistance of each component and other parts of the circuit.
2. Determine the voltage across each component and other parts of the circuit.
3. Determine the current through each component and other parts of the circuit.
4. Fault any of the resistors in the circuit and take more measurements.

Table 4.1 Set of tasks

None of the four subjects who took part in this study were given any training about how to use the interface. Subjects were free to explore the environment and if, after a period of time, not exceeding 10 minutes, they were uncertain about any aspect of the interface information would be provided to help them.

4.5 Results

Subjects' mouse-clicking actions were analyzed with respect to their verbal reports with a view to establishing the degree of correspondency between them, in particular, anaphora, ellipsis and deixis. The results of the study are presented below.

4.5.1 Graphical equivalents of natural language

The information in the following excerpts is presented in two ways. Combinations of icons highlighted by a subject through their mouse selections form the graphical dialogue, and any utterances accompanying these selections form the natural language dialogue. It is suggested that the graphical actions performed by subjects are in some sense analogous to these utterances and therefore are a reflection of the information currently being attended to.

Excerpt 1 from subjects 1, 2, 3 and 4 indicates the information they are currently attending to, which is being verbalised as they graphically interact with Circuit I.

Excerpt 1

	Graphical Dialogue			Natural Language Dialogue
• Subject 1				
	Meter	Component Selected	Result Value	
12.	Ohms	R1	3.0Ω	Resistance of R1 is 3 ohms.
13.		R2	6.0Ω	R2 is 6.0 ohms.
14.		R3	1.0Ω	R3 is 1 ohm.
15.		Bulb	5.0Ω	the bulb was 5 ohms.
16.			0Ω	and the source is 0.0 ohms.
17.	Volts	P1 P4	1.5v	The voltage across P1 and
18.				and P4 is 1.5 volts.
19.		P4 P5	0.75v	across P4 and P5 is 0.75 volts.
20.		P5 P6	3.75v	across P5 and P6 is 3.75 volts.
21.		Bulb	3.75v	3.75 volts.
22.	Amps	Bulb	0.75A	I can do currents: the bulb
23.		R3	0.75A	and R3 both have 0.75 amps.
24.		R2	0.25A	R2 has got 0.25 amps, and
25.		R1	0.5A	R1 presumably .5 which it
26.				does, which is consistent.

The ability of subject 1 to associate different words with his mouse-clicking actions provides him with the freedom to express his intentions to Circuit I, not only in a short-hand form, but in a range of ways that would probably be unavailable in current natural language interfaces.

• **Subject 2**

	Meter	Component Selected	Result Value	
27.	Volts ◎	P2 ● P4 ●	1.5v	Let's measure some
28				voltages.
29.		P1 ● P2 ●	0.0v	Do I get nothing if I read the
30.				voltage across there... right,
31.				so the leads have no voltage
32.				drop across them.
33.		R3 ⊣▭⊢	0.75v	The voltage across R3
34.				The voltage across the
35.				bulb...
36.		Bulb ◉	3.75v	Now if I apply Kirchoff's
37.				laws to that I get 3.75 volts.

The variation of dialogue generated for similar mouse-clicking actions (in this case ellipsis) can be seen by comparing subject 1's actions (statements 19 - 21) and subject's 2's (statements 29 - 36).

• **Subject 3**

	Meter	Component Selected	Result Value	
38.	Volts ◎	P1 ● P4 ●	1.5v	I've got the voltage across
39.				there which is 1.5 volts
40.				across the pair.
41.		R3 ⊣▭⊢	0.75v	I've got 0.75 across the
42.				resistor R3,
43.		P1 ● P5 ●	2.25v	2.25 volts across the resistor
44.				chain, and 6.0 volts

45.			altogether and that gives
46.			3.75 volts (across bulb)... is
47.			that right... let me check...
48.	Bulb	3.75v	how much across the bulb?

This variation can be seen again where subject 3 expresses his intentions elliptically (statements 41 - 48) in exactly the same way as subject 1 (statements 19 - 21).

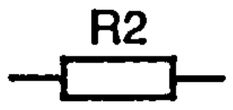
- **Subject 4**

	Meter	Component Selected	Result Value	
49.	Ohms	R1	3.0Ω	R1 is 3.0 ohms.
50.		R2	6.0Ω	R2 is 6.0 ohms.
51.		R3	1.0Ω	R3 is 1.0 ohm
52.		Bulb	5.0Ω	and the bulb is 5.0 ohms.
53.		P1 P4	2.0Ω	The resistance of R1 in
54.				parallel with R2, and
55.		P1 P5	3.0Ω	R1 in parallel with R2 plus
56.				R3 is 3.0 ohms, and the total
57.		P1 P7	8.0Ω	resistance of the circuit is
58.				8.0 ohms.
59.	Volts	P1 P4	1.5v	The voltage across R1 and
60.				R2 is 1.5 volts.
61.		R3	0.75v	R3 is 0.75 volts.
62.		Bulb	3.75v	The bulb is 3.75 volts,
63.	Amps	Bulb	0.75A	its current is 0.75 amps..

64. 

0.5A The current through R1 is

65. 0.5 amps.

66. 

0.25A R2... 0.25 amps.

67. 

0.75A R3... 0.75 amps.

68. 

6.0v The total voltage is 6.0 volts.

69. that looks OK.

The advantage of allowing users to express themselves in a graphical short-hand form is that the meaning of their actions is context dependent. For example, changing the specified use from Volts to Amps whilst referring to the same object (bulb) allows subject 1 (statements 21 - 22) and subject 4 (statements 62 - 63) to mean something quite different.

4.5.2 Graphical expression of anaphora and ellipsis

One of the main goals of this research was to determine if linguistic phenomena (anaphora and ellipsis), which are present in natural language, could be modelled graphically. The importance of such phenomena in human communication is demonstrated by the apparent ease with which conversants use them to track a conversation as it proceeds.

Analysis of subject protocols and their corresponding mouse-clicking actions have shown that analogues of anaphora and ellipsis can be expressed graphically and are not phenomena exclusive to natural language. Subjects vary a command (anaphora) or argument (ellipsis) portion of a command-argument or argument-command syntax, via mouse selections, to express these phenomena.

Burton et al. (1979) give the following definition of natural language anaphora and ellipsis.

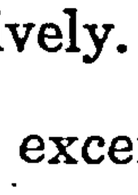
> *"If the problem of pronoun resolution [anaphora] is looked upon as finding a previously mentioned object for a currently specified use,*

then the problem of ellipsis can be thought of as finding a previously mentioned use for a currently specified object."

Applying these definitions to the graphical dialogue, it can be seen that subject 1 has made mouse selections in statements 13 - 16 which are analogous to natural language ellipsis, where the arguments R2, R3, Bulb and Battery, are the currently specified objects looking for a previously specified use, which in this case is the command 'measure ohms'. Other examples of 'graphical' ellipsis can be found in statements 19 - 21, 23 - 25, 29 - 36, 41 - 48, 50 - 57, 61 - 62 and 64 - 68 of excerpts 1, 2, 3, and 4 which correspond to their natural language counterpart.

Mouse selections analogous to natural language anaphora are shown in statements 21 - 22, 62 - 63 of excerpt 1 and 75-76 of excerpt 2 where the command 'measure amps' is the currently specified use looking for previously specified object, which in this case are the arguments bulb and R3 respectively. Subject 4 also expressed a 'graphical' anaphor in statements 80 - 81 of excerpt 2 where the command 'measure volts' was the currently specified use, and the previously specified object was the argument Bulb.

Excerpt 2

	Graphical Dialogue		Natural Language Dialogue
	• Subject 1		
	Meter	Component Selected / Result Value	
70.	Ohms 		Well I might as well try
71.			Ohms because that's what
72.			you ask me.
73.			So it says with R3 that the
74.			resistance is 1.0 ohm and

75.	Volts ◉	0.75v	the voltage is 0.75
76.	Amps ◉	0.75A	and the current is 0.75
77.			which seems to confirm
78.			that the multimeter is
79.			connected to that resistor.

• **Subject 4**

	Meter	Component Selected	Result Value	
80.	Amps ◉	Bulb ◉	0.75A	So the bulb is 0.75 amps
81.	Volts ◉		3.75v	and voltage of 3.75 volts.

It is interesting to note that none of the natural language dialogue associated with subjects' graphical anaphors contain any pronouns (for example, 'it', 'that', 'these') which pertains to previously specified objects. These results show that Circuit I supports ellipsis, which seems to perform the function that in verbal conversation may be done by either anaphora or ellipsis.

4.5.3 Graphical expression of deixis

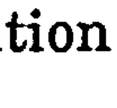
Deixis within an interface, like its natural language counterpart, allows users to point whilst talking. It subsumes those features of language which refer directly to the temporal or locational characteristics of the situation within which an utterance takes place, whose meaning is thus relative to that situation, for example, now/then, here/there, I/you, this/that are 'deictics' (Crystal, 1980). Deixis within multi-modal interfaces "...allows users to take advantage of the graphical environment of the display to reduce the verbosity of their natural language input, increase the specificity of their requests, and increase the naturalness of the interaction." (Cohen et al.,

1989). These benefits also apply to Circuit I which seeks to exploit the inherent linguistic properties of graphical objects without resorting to natural language. Examples of deixis occurring in subject protocols are illustrated in statements 82, 86-89 and 92 of excerpt 3.

Excerpt 3

	Graphical Dialogue				Natural Language Dialogue
	• **Subject 1**				
	Meter	**Component Selected**		**Result Value**	
82.	Ohms 	P1 ●	P4 ●	2.0Ω	If I go **there** and **there** ... I
83.					can measure resistance
84.					from any two points ... that
85.					gives me 2 ohms.
	• **Subject 2**				
	Meter	**Component Selected**		**Result Value**	
86.	Ohms 	P3 ●	P4 ●	6.0Ω	I've got 6 ohms across **there**,
87.		P2 ●	P5 ●	7.0Ω	7 ohms **there** because
88.		R3 		1.0Ω	that's still 1 ohm, right, and
89.		P5 ●	P6 ●	5.0Ω	**that** reads 5 ohms as it read
90.					before.

- **Subject 3**

	Meter	Component Selected	Result Value	
91.	Ohms 	P1 ● P4 ●	2.0Ω	Now I'm looking at R1 and
92.				R2 and **that** is 2 ohms.

Further analysis of subjects' graphical dialogue indicated that their mouse-clicking actions associated with deixis were identical to those used to express graphical ellipsis, the only distinguishing factor being the natural language utterances they associated with such actions.

Given that concurrent verbal protocols are a direct reflection of what information subjects are currently attending to, and depending upon the task complexity, their mouse-clicking actions may be expressed deictically in conjunction with either verbal or non-verbal communication. It should be noted that in direct manipulation all communication is by deixis, so in this respect Circuit I is providing nothing new to users.

4.5.4 Analysis of subject protocols

Each subject's protocol was further analyzed, in respect of general comments made, in an attempt to establish their overall impressions of the interface and changes they would like to see in future versions of the Circuit I.

- **Subject 1**

Although there was no direct link between the multimeter and the circuit, S1 was able to conclude that by highlighting a multimeter icon (for example, Ohms button) and one of the circuit's components the appropriate value would be given.

There was a distinct lack of tools provided to use on the circuit. S1 felt that if we were going to be looking at electric circuits he would want to do a lot more than was available in this interface. He thought it was rather limited and did not seem intuitive to use.

S1 held the view that he did not think about circuits in natural language but rather he thought about them graphically. When asked if he thought about how the circuit functioned in graphical terms, he said that it rather depended on what he was trying to work out. Intuitive thoughts about circuit functions would be thought about graphically, but when looking at, for example, the parallel combination of resistors then he might well be thinking about a calculation.

S1 views his own methods of working with circuits as primarily qualitative, but quantitative in the sense of having to sketch out the particular calculation he wants to do and have the multimeter compute the actual calculation. As he said "Because I had a computer there and because I had a multimeter, I avoided doing mental calculations. Where I did calculations I jotted them down, but where possible I tried to get the multimeter to do the work."

S1 also found that sometimes the inspection points that he expected to be highlighted were not and this meant going through an extra step he hadn't expected to do. He found this surprising rather than annoying, although most of these problems he attributed to HyperCard itself.

After measuring, for example, the voltage across two points, P2 and P6, and then selecting P4 to find that the voltage across P4 and P6 was given, S1 reported that he wasn't aware of what was happening. As he said, "just sometimes nodes didn't do quite what you expected, but I was able to gain control so it wasn't a limitation."

Since the circuit's current layout avoided such issues as Kirchoff's laws by only having only two-way junctions, S1 was asked if he preferred this layout as opposed to a printed circuit board one. After some thought S1 said that he had no preference.

The voltage source did not follow laws that S1 had expected, but this was because he misunderstood the operation of the multimeter. The bulb however, caused S1 some uneasiness because he could not establish if it was a linear or non-linear device. He felt that some information clarifying this would be helpful.

- **Subject 2**

S2 found that the interface of Circuit I was easy to use once its different parts and functions were explained to him. He thought that the circuit's components were fairly well represented by the interface.

Measuring voltage across two inspection points seemed natural enough to him, but measuring current through a node (having only two connections) did not, as in a real circuit you would have to break the circuit to do this. Taking resistance measurements without removing the battery from the circuit also caused him some problems because in a real situation he would remove the battery. However, he could accept these things because he was dealing with a computer simulation.

S2 had no problems in obtaining values of the different parts of the circuit. Rather than using the dialogue and faults boxes to elicit more information about the state of the circuit, S2 took a "fairly naive approach" by "just measuring all the voltages and currents... and then just working out the rest without using the power of the circuit...".

Whilst checking different values throughout the circuit, S2 formulated a range of questions in his mind. As S2 explained, "What was going through

my mind was... does this circuit obey the basic laws... Kirchoff laws? do all the voltages across the parallel combination of R1, R2, R3 and across the bulb add up to 6.0 volts, add up to the source? ...does the same current flow into the parallel combination as flows out through the bulb and through R3?... so the basic laws are maintained. At the same time, do the readings that I get by putting the multimeter across R1 and R2 and measuring voltage, and current, do they square up with my expectations of Ohms law? Those were the things going through my mind." When asked if he posed questions to himself about, for example, the voltage across two inspection points, or the resistance of a component he said "Yes, I certainly was doing that with the aim of finding out whether those measurements were consistent with how I thought the circuit should behave."

Clicking on a component to take a measurement, as opposed to clicking on inspection points, seemed to be more intuitive to S2 although clicking on an inspection point to get its current was not.

When highlighting two inspection points and subsequently changing one in favour of another, S2 found it mildly irritating when it did not give him the measurement he had wanted. However, he got used to this and by choosing his point carefully he could systematically step through the circuit. Tracking users' mouse selections when only one component or part (representing an argument) is being varied presents no problem for Circuit I. However, in the case of two arguments, when taking a voltage measurement across two inspection points, varying one inspection point in favour of another causes an ambiguous situation to occur. Circuit I has to choose which one of the inspection points to de-highlight and with the current mechanism supported (first highlighted, first de-highlighted) it has only a fifty percent chance of interpreting the users' intentions correctly. In those cases where it does not de-highlight the inspection point intended by a user it fails to mirror their intentions.

As far as the circuit board layout was concerned, S2 preferred the one shown by Circuit I, as opposed to one displayed from a printed circuit board perspective. S2 uses a theoretical model of a circuit (formulae describing its behaviour) and a physical layout (like the display of Circuit I) to relate to and understand circuit behaviour.

S2 would like to see the 'Reset' box done away with and perhaps the reset button being placed within the multimeter. In its current position, S2 thought that the reset button was completely disconnected from the task it performed.

- **Subject 3**

S3 commenced by querying the use of the power supply and had to be told that the battery was removed from the circuit whilst resistance measurements were being taken. From this he presumed that when measuring the current through a component the battery was put back into the circuit.

S3 had some difficulty in getting to grips with the interface which meant that he had to stop and ask whether what he wanted to do was possible. These interruptions led to his 'train of thought' being interrupted.

After a few calculations S3 realized that the battery had no internal resistance, and that the current in each 'arm' of the circuit could be measured. However, S3 thought that the current could be measured by selecting two different inspection points until it was pointed out to him that one inspection point was sufficient.

When asked what he thought of the interface, he felt that it would be nice to get a practice run in first to gain some familiarity in using it.

S3 also found the nearness of the multimeter buttons (amps, volts, and

ohms) confusing and thought that they might be better placed in a horizontal pattern than in a vertical one.

Upon reflection S3 found the interface intuitive to use once he got over his initial difficulties. Having introduced a fault into the circuit and taken a voltage reading across a component, he would have liked to have been able to clear the fault(s) in one action and obtain the correct reading across that component.

S3 said that he had no difficulty in ascertaining what the state of the various components were. Although he was shown that a component could be selected (as opposed to the inspection points across it) he chose not to use this facility. He could offer no reason for this.

When asked if he formulated mental questions in his head when taking measurements, S3 said yes, "but mainly at the level of what I was asking." S3 felt that if the circuit were more complex he would begin to ask himself more mental questions.

The use of a 'scratch pad', suggested S3, might be a useful addition to the circuit for displaying information already obtained from previous interactions although a larger screen display would be needed.

- **Subject 4**

S4 said that he liked the interface and found it intuitive to use. One misconception that he had was to think that the current could be measured between two points. It seemed to S4 that clicking on components overcomes many of the problems incurred when clicking on inspection points.

As far as the 'Reset' button was concerned, S4 said that it seemed out of context in its present position and should be removed and placed perhaps within the multimeter. This, he said, would be a more logical place since

clicking on it clears the multimeter display and resets the highlighted points.

S4 found the circuit layout acceptable because he normally holds a schematic version of a circuit in his head as opposed to a printed circuit version which he finds unnatural.

When S4 was asked if the verbal reports he was making were a consequence of formulating mental questions in his head about the circuit, he said he was unsure. S4 thought that what probably governed his thinking was the way questions were asked about resistance, current and voltage. If he had been left to play around with the simulation he might have treated it differently, instead he treated it as a set of linear questions. As S4 said "I was just answering the questions... if it was more open then I would have treated the components separately and gone round the diagram."

S4 suggested that instead of being given all the circuit values it would be better if there were somewhere to store them. For example, if a resistance measurement is taken, this plus the current and voltage measurements could appear in the dialogue box. Also, when a component is faulted its specification could be displayed within the dialogue box on one side and the voltage, current and resistance values on the other. Such a facility, he suggested, would be very useful because it would enable a comparison of the two sets of information.

4.5.5 Summary of subject protocols

In addition to analysing recorded protocols for anaphoric, elliptical and deictic actions, a summary of subjects' comments concerning their interaction with Circuit I was compiled.

Three of the four subjects questioned the use of the multimeter because there seemed to be no apparent direct link between it and the circuit. Once its

operation was explained subjects quickly accepted the fact that they could use it to measure values (voltage, current or resistance) by merely pointing to and highlighting either the inspection points or the various components.

Subjects also felt that it was not obvious that the voltage source was removed from the circuit whilst resistance readings were being taken. Subject 2 felt this could have been made more explicit, as he said "I came to terms with it because it was a computer simulation." On the whole all subjects would have preferred to see it removed from the circuit before measuring resistance, but could get used to it remaining in place. As subject 2 pointed out "I don't think that it's insuperable, but it's a difference."

The bulb's apparent lack of a model caused subjects 1 and 2 to question its voltage-current relationship. They both felt that some information clarifying this would be helpful.

Subjects 1 and 2 also felt that not having the particular node that they wished highlighted was mildly irritating. As subject 2 said "I think that I got used to that because I felt that if I chose my points correctly I could click systematically through a circuit." He also preferred to click on components as opposed to the inspection points because:

S: 7. Clicking on the components actually overcomes one of the
8. problems I mentioned about the intuitive use of the multimeter
9. with current, in that clicking on the components perhaps
10. includes within it the notion that you have the 'voltage across'
11. and 'current through [it]', that's what a component is, that's
12. actually part of its definition, whereas just clicking on one blob
13. and having to say to myself "I must remember that I'm
14. measuring the current past that point" is perhaps an uneasy fit.

In addition, it was found that highlighting components in this way

overcame the problem of Kirchoff's current law which three-way junctions give rise to, i.e., the algebraic sum of the currents in amperes at any junction at any instance equals zero. Subjects 3 and 4 experienced no problems in ascertaining the different states of components. Subject 3 chose not to highlight components to ascertain their states, but other than forgetting, could not account for not using this facility.

Since Circuit I's presentation avoided the issue of Kirchoff's current law by only having two way junctions, subjects were asked if they preferred this layout as opposed to one displayed from a circuit board perspective. Subject 1 said that he had no preference, and subject 3 thought that whilst the layout was fine he did wonder how far you would want to go with this approach. Subject 2 liked the layout and found that it corresponded to the way he thought about circuits, as he said "I find printed circuits tell me nothing about the operation of the circuit. It's just a load of components laid up in a way which aids the construction of the circuit board." Subject 3, who believed that he normally has a schematic version of the circuit in his head, agreed with this conclusion.

The 'Reset' icon which clears the highlighted components and multimeter display caused some confusion to subjects 2 and 3. Both subjects had misconceptions of what it did. When its real function became apparent they both felt that it should be placed within the multimeter display.

The use of pull-down menus to fault and clear components, and to obtain a component's specification seemed unnatural to subjects 2 and 3. Both would have preferred to fault and re-fault a component directly rather than using such menus.

Having to take so many readings of the various components, all subjects noted them down on paper and used them as a means of cross-checking against their own calculations. Subjects 3 and 4 both mentioned that some

sort of 'scratch pad' would be useful so that when a component was faulted its specification could be displayed on one side whilst the various readings taken up to that point could be displayed on the other.

4.6 Summary

The empirical study conducted with Circuit I has shown that linguistic analogy has a role to play within a graphical environment. A comparison of subjects' natural language dialogue against their graphical actions seems to suggest some degree of correspondence between the two in the sense that they were formulating questions in their mind to pose through their mouse selections. This was a view not supported by the subjects as only one said that he was formulating questions as he clicked on various objects with the mouse. However, subjects frequently varied the argument and command portions of the command-argument or argument-command syntax supported by Circuit I. This seems to suggest that, when supported, users' mouse selections which perform analogous roles to anaphora and ellipsis are used in a similar manner to their natural language counterparts.

One of the conclusions to be drawn from the results of the protocol analysis is that supporting free-order syntax greatly enhances the usability of directly manipulation interfaces. The claim is supported by the protocol extracts which show how easily commands and arguments representing anaphora and ellipsis may be expressed and modified. Rather than being considered narrow linguistic features, anaphora and ellipsis support extended interaction rather than isolated propositions. Supporting analogues to these phenomena within a direct manipulation interface provides Circuit I's users with the same degree of brevity, naturalness and convenience experienced by users of SOPHIE.

As discussed in §1.1, all direct manipulation interfaces support deixis.

However, all pointing is ambiguous and you have to combine the direction of pointing with other information to decide what is meant (Draper, 1986). Circuit I, like many other direct manipulation interfaces is limited in this respect also, which is inflexible. To enhance the usability of Circuit I a more flexible form of deixis needs to be supported which more closely approximates its natural language counterpart. That is, the ability to point and mean more than one thing. The improved version of Circuit I, Circuit II will seek to support this.

Another limitation of Circuit I is its inability to support topic shifts between two contexts, normal and faulted circuits. When Circuit I is modified the previous context, i.e., the state of the normal circuit context, is not remembered. When the normal circuit context is restored the previous state has to be set up again. This limitation will be addressed in Circuit II.

Chapter 5 and 6 discuss the design and implementation of a fully developed model, Circuit II, which extends the work of Circuit I.

Circuit II - internal design

5.1 Introduction

This chapter describes the internal design of Circuit II which represents a re-implementation of the Intelligent Tutoring System SOPHIE, a powerful reactive learning environment within the domain of electronic troubleshooting.

5.1.1 Objectives and background

It has long been established that simulation models can be used as platforms upon which to teach and refine students ideas on qualitative physics (Wenger, 1987). Many systems developed have contributed to this field, but one such system, SOPHIE, stands out from the rest.

SOPHIE is now considered a 'milestone' in the evolution of Intelligent Tutoring Systems. Its simulation specialists analysed and manipulated circuit knowledge and the delivery of that knowledge through its natural language capabilities made it a very powerful instructional system. Inspired by the work on SOPHIE, the model presented in this chapter, Circuit II, an extension to Circuit I presented in §3.0, is a re-implementation of some of the different computational parts of SOPHIE and their operation.

The main motivation behind this model has been the desire to reconstruct SOPHIE in a modern context. As already discussed (see §2.0), SOPHIE's powerful natural language capability enabled it to appear 'human-like' in terms of the responses it gave to student-posed queries. Since eighteen years have passed since SOPHIE has been implemented, many Intelligent Tutoring Systems have developed graphical interfaces as a communication

medium because of the difficulties yet to be resolved in natural language interfaces. However, prior to this study, these interfaces had paid little attention to the issues of anaphora, ellipsis and conversational digressions with a graphical interface. Circuit II has been developed to establish if a graphical interface can match and extend SOPHIE's natural language capabilities. In order to do this many of SOPHIE's computational modules have been re-implemented in Circuit II. This chapter describes these modules.

5.1.2 Goals

Circuit II, like SOPHIE, has to be able to understand how the circuit it models functions so that when user-questions are posed it can respond to them intelligently. This means that Circuit II has to be able to accept meaurement, hypothetical and random fault types of questions about any component modelled in the circuit. It then has to be able to compute the values, and depending upon the particular context in which the question has been asked, respond appropriately. In the case of randomly inserted faults, Circuit II has to be able to critique hypotheses presented by users which are inconsistent with a set of measurements observed.

In a similar manner to SOPHIE, Circuit II must respond in a 'friendly' manner and support direct manipulation techniques which allow users to communicate with it. The direct manipulation environment must support two other features modelled by SOPHIE, namely, anaphora, and ellipsis.

Another goal in constructing Circuit II is to try and imitate the apparently effortless shifts that people make when moving between topics. Rather than clicking on, for example, windows or issuing explicit commands to

reactivate or close windows, Circuit II must support automatic shifts of topic made by users.

The last goal in constructing Circuit II was to support a more general form of deixis, a feature not supported by SOPHIE, within direct manipulation style design. To extend the usability of SOPHIE, the ability to point and mean more than one thing must be supported.

5.1.3 Performance

Circuit II has achieved all these goals and is capable of interpreting and responding to user input extremely quickly. The time taken by Circuit II to parse graphical input, semantically interpret it and provide an answer in any context takes about one second. Inserting either a hypothetical or random fault takes two seconds on a Archimedes A5000 running RISC OS 3.0. These figures compare favourably with SOPHIE's performance which took approximately the same time to perform similar tasks on a PDP-10 running TENEX.

5.1.4 Implementation

Unlike SOPHIE, Circuit II is implemented entirely in BASIC VI and includes a general purpose circuit simulator system, also written in BASIC VI. The Total size of the program is 385k, 26k of which is associated with the circuit simulator. The simulator currently only models the audio-video amplifier circuit which is reasonably complex containing twelve resistors, six capacitors, three transistors, a D.C. power supply and video output.

The graphical interface is stored in a file containing 1Mb of 'sprites' representing the individual parts of the circuit whose manipulation is

controlled by various parts of the BASIC program. The interface of Circuit II has been developed on a high-quality bitmapped colour display with a resolution of 1056 by 256 pixels.

The primary goal in implementing Circuit II was that the dialogue modelled be as rich and complex as SOPHIE's. Tutoring students in electronic troubleshooting was not a major consideration. Circuit II is a fairly full re-implementation of SOPHIE I with the exception that a) it does not model fault propagation b) it does not store a user's knowledge about a fault and c) the model cannot generate hypotheses about the nature of a fault. Circuit II's capabilities could be extended to model these features making it suitable as an electronic troubleshooting tutoring system. Apart from these deficiencies, Circuit II is of equal complexity to SOPHIE I and is capable of performing as well in all other areas.

A full description of the modules used by Circuit II, their functions through which various specialists and the general purpose circuit simulator work is given in appendix I.

5.2 Graphical processor

A very important feature of SOPHIE was the natural language capabilities of its interface. The use of semantic grammar using augmented transition networks enabled it to decompose an input sentence into categories associated with the domain of electronics. For example, the question "What is the voltage across R1?" would be parsed by SOPHIE's <REQUEST> function which would then decide that this statement is a request for information. The sentence would then be recursively broken down into its

constituent parts, i.e., measurable quantity and location, before being evaluated to answer the student's question.

In a similar manner Circuit II exploits the constrained nature of the electronics domain to answer questions posed to it by allowing students to interact directly with the graphical representation of the circuit on the screen. The parser is not so complex as SOPHIE's but just as efficient and capable of handling a wide range of iconic input.

5.2.1 Description of the parsing process

Circuit II's interface differs from SOPHIE's in that natural language is not used as an input medium. All input to Circuit II is mouse-driven which enables students to view the mouse pointer as a probe, as in the case of an electrical meter probe. Since many measurements taken in electronic circuits are done with instruments using probes it was thought that this method of input might be more appropriate. Like a real probe the mouse pointer is used to select icons of circuit components representing arguments, and commands, for example, voltage and/or faults which represent actions. Each icon represented in Circuit II has an integer value associated with it, such that, when the icon is activated that value is assigned to an array representing either a command or argument. Iconic input can either be (a) a request for information or (b) a modification to be made to the circuit. To determine if the input is a request for information, Circuit II checks that both arrays Use and Comp associated with the semantic category of *measurement* have been instantiated. If a request is being made then the values of these arrays are used to reference the appropriate table of values, for example, Voltage, to retrieve the measurement. If the statement is not a request for information then Circuit II checks the arrays Fault and Hyp

associated with the semantic category *modification* to see if changes to the circuit model are required. If so, then the circuit simulator is invoked to update the reference tables based on the requested modifications.

5.2.2 The parser

When two icons associated with a particular semantic category (i.e., measurement or modification) are activated the library corresponding to the component type (capacitor, resistor or transistor) is referred to. The activated component and its semantic category are identified by a) its x and y coordinates, b) the mouse button used to activated the component (1 - measurements, 2 - modifications, 3 - specifications) and c) the number (1 - volts, 2 - current and 3 - resistance) returned by the Use or Fault array. If a measurement is requested then the meter, component and ellipsis arrays, Use, Comp and Ellip, within the component and meter libraries are instantiated with values corresponding to the activated icons. Should the Ellip array already contain a value from a previous measurement then the procedure Ellipsis is invoked which de-highlights the previously activated component in favour of the new one. The numerical value held in the array Use determines which table of values is referenced for the current measurement being taken and is handled by the procedure Meter within the meter library. If the value of Use changes, for example, from voltage to current when a component is highlighted, then the procedure Specified_Use is invoked to re-interpret the iconic input in the light of this change. When both arrays Use and Comp are initially instantiated with values or are updated, the measurement library is referred to which displays an appropriate response.

In the case of circuit modifications, the value held in the array Fault signifies the category of fault (for example, 4 - high resistance, 7 - short-circuit) and the arrays Hyp and Ellip2 identify the component to which the fault is to be applied. When a component is faulted the procedure Fault_Ellipsis is invoked. If the array Ellip2 already contains a value this is used to identify and de-highlight the previously faulted component in favour of the newly inserted one. The procedure corresponding to the inserted fault, for example, (short, open), held in numerical form in the array Fault, is invoked, which modifies the circuit model (Write_Spice) before the circuit simulator is called to compute new values for the tables of reference. If the type of fault applied to a component is changed then the procedure Fault_Use is invoked, which modifies the circuit model before computing new circuit values. As well as visually reflecting that a component has been faulted, a message is also displayed to that effect.

When a component's specification is requested by pressing the third mouse button over it, the procedure Spec_Check is invoked which uses the x and y coordinates of the component to retrieve and display its specification.

As discussed, Circuit II's free-order syntax allows the student's input to vary (a) the component across which a measurement is to be taken, (b) the type of measurement to be applied to a component, for example, voltage, current, (c) the component to which a modification is to be made and (d) the type of modification to be applied to a component. Once a question has been posed to Circuit II these categories of variation provide students with a very flexible way of modifying parts of the iconic input by either varying the argument portion ((a) and/or (c)) or the command portion ((b) and/or (d)) of the syntax. The advantage of this type of free-order syntax over SOPHIE's natural language processor is that it eliminates ambiguity about what is

being referred to because of the explicit nature of iconic input. By contrast, when SOPHIE received input which was not completely specified or ambiguous it did not always return the expected answer. This was a major limitation of SOPHIE's natural language processor.

5.2.3 Tracking conversational moves

As well as supporting argument and command replacements in the iconic input, the parser can also determine when a *conversational move* to another context is signalled. The graphical processor currently supports three different circuit contexts: (a) normal circuit behaviour, (b) a circuit with an unknown fault and c) circuits with user-hypothesised faults. To move between these separate but related contexts SOPHIE provided students with commands SAVE to save a current context and RESTORE to return to it. Within the context of a direct manipulation environment Circuit II provides graphical equivalents of these commands called *cue phrases* to signal that a conversational move to a different context is desired.

Reichman (1985) has described many different types of cue phrases which occur naturally in conversation to signal different types of conversational moves. For example, the cue phrases 'Incidentally...' or 'By the way...' may be associated with the interruption and suspension of a currently active context, and 'Anyway...' or 'In any case...' with the resumption of a previously suspended context. The commands SAVE (an interruption) and RESTORE (a resumption) may be thought of as cue phrases too which perform the same functions. The provision of these cue phrases in Circuit II provide students with an iconic means of digressing between contexts when desired.

When a student wishes to fault a component, Circuit II's syntax will allow the type of fault (for example, short, open), and component to which it is to be applied, to be specified in any order using the second mouse button. Within a normal working circuit environment, when the second mouse button is pressed over a component or fault icon, the procedure ContextCheck (in the CxtLibrary) is invoked which determines that a digression to a hypothetical fault context is about to occur. This in turn invokes other procedures, PROCSaveContextSpaceC1Status and ChangeContextC1ToC2 (both in the CxtLibrary), which save the current context, reset the icons and change the window border colour from green to red signifying a faulted context. In this context hypothetical questions of the "If x then y" variety can be posed. When a student wishes to resume the previously interrupted context (i.e., in the normal working circuit) they do so by placing the mouse pointer over a label "norm circuit" and pressing it once with the first mouse button. This action again invokes the procedure ContextCheck which determines that the currently suspended context should be reactivated. It does this by invoking procedures ChangeContextC2ToC1 and PROCRestoreContextSpaceC1Status (both in the CxtLibrary) which changes the window border colour from red to green and highlights the icons used previously in this context. Any measurement taken prior to the digression will be re-displayed with a 'return' clue word phrase, for example, "Anyway, as I was saying, the voltage across here is 7.5290 volts."

If a student wishes to troubleshoot the circuit they may activate the icon which signifies the insertion of a random fault with the second mouse button. Activating this icon invokes the procedure ContextCheck which changes the context from a non-faulted context to one in which a random fault has been inserted. The insertion of a random fault is not considered a

conversational digression. At present, whilst students can present hypotheses to Circuit II about what the fault might be they cannot make hypothetical modifications to the circuit in this context to test them out. This is another feature of SOPHIE not modelled in the current implementation of Circuit II.

5.2.4 The range of SOPHIE's linguistic capabilities modelled by Circuit II

The free-order syntax supported by Circuit II allows students to pose a wide range of questions in their equivalent iconic form which are analogous to that supported by SOPHIE's natural language processor. Listed below are examples of students' questions which are accepted by SOPHIE and which can be presented to Circuit II iconically.

Requesting measurements

What is the voltage across the base emitter junction of Q1?

What is the VBE of Q3?

What is the current through the base of Q2?

What is IB of Q1?

What is the output voltage?

What is the voltage between node 1 and the positive terminal of node 14?

What is the resistance of the 4k7 ohm resistor?

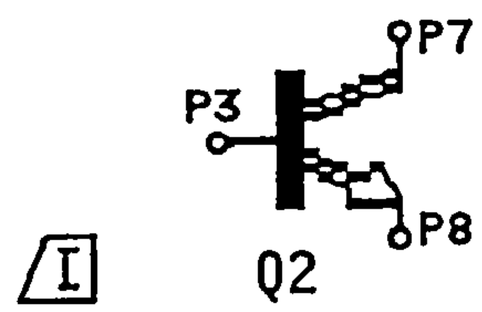
What is the specs of Q3?

In a working circuit what is the d.c. input voltage?

Example of a measurement request

question:

iconic representation



What is the current through the base of Q2?

Modifying the instrument

Make the load resistor go high.

Make the base emitter of Q1 go open.

Set the 1k2 resistor to a high value.

Short the collector-emitter of Q2.

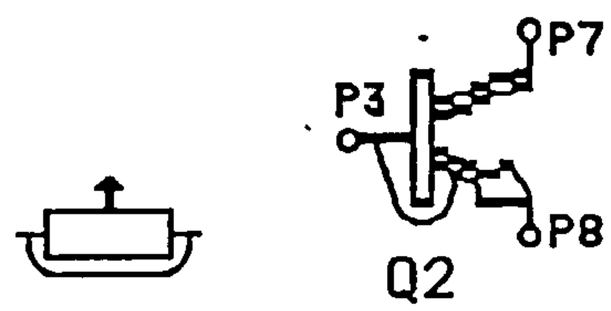
Short circuit the base-emitter of Q2.

Example of modification request

request:

iconic representation

Short circuit the base-emitter of Q2



Questions:

What happens to the VBE of Q3 when the 470 ohm resistor shorts?

If the 1k2 resistor opens then what happens to the voltage between nodes 1 and 6?

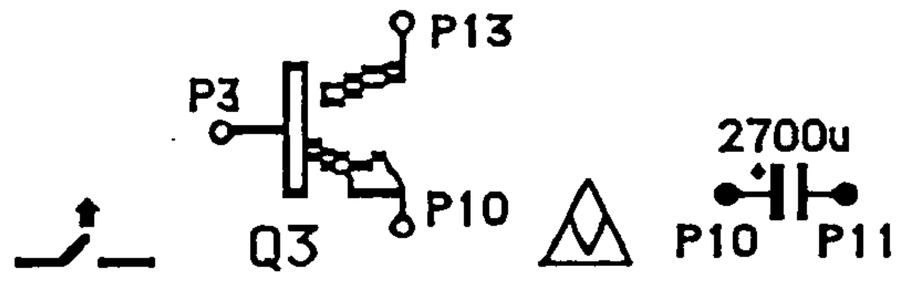
What happens to the voltage across the 2700 microfarad capacitor if the base-collector of Q3 opens?

Example of a question

question:

What happens to the voltage across the 2700 microfarad capacitor if the base collector of Q3 opens?

iconic representation



Hypothesis checking

Is it possible that the base-emitter of Q2 is shorted?

Could the problem be that the 1k2 resistor is open circuit?

Could the 4k7 resistor be high?

Replace Q2.

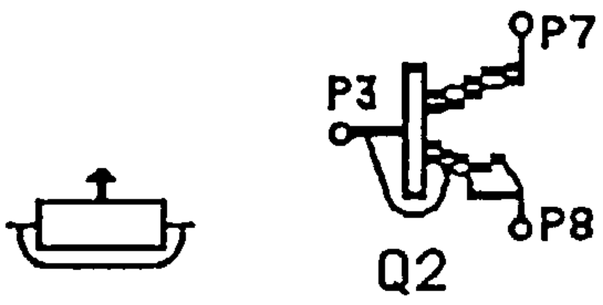
Replace the 82 ohm resistor.

Example of hypothesis checking

question:

Is it possible that the base-emitter of Q2 is shorted?

iconic representation



Elliptical statements:

Measurements:

Through the base-emitter of Q3?

across the 1k2?

470?

Faults:

Short the 82 ohm.

1k2.

Open the base-emitter junction of Q2.

base-collector.

Example of ellipsis

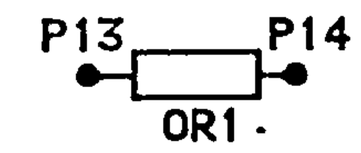
question:

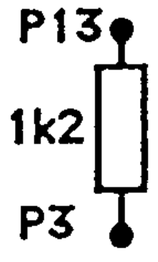
What is the voltage across OR1?

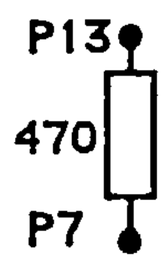
across the 1k2?

470?

iconic representation







Anaphoric-like statements

Measurements:

Is that right?

Is that wrong?

What's its current?

Voltage?

Faults:

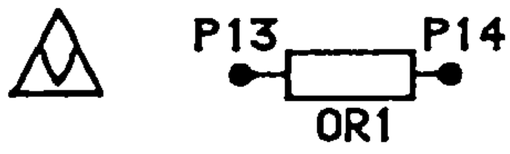
Short it.

Open it.

Make it go high.

Make it go low.

Example of an anaphoric-like statements

question:	**iconic representation**
What is the voltage across OR1?	
How about its current?	
Is that right?	

Miscellaneous commands

Remove all faults

Save

Restore

Reset the instrument

Example of a miscellaneous command

question: **iconic representation**

Restore 

5.3 Device Specialists

Being able to change the context of the circuit enables users to explore three different scenarios i.e., normal working, hypothetical and random fault environments. Circuit II provides users with a number of specialists which enable them, for example, to take measurements, ask factual questions, fault and replace components. In this section each of the specialists are discussed providing details of what they do and how they achieve it.

5.3.1 Measurement specialists

Three specialists are used to calculate voltage, current and resistance measurements across resistors, capacitors, transistors, between two points in the circuit or at a single point with reference to ground. To access, for example, the Voltage specialist a user must select the voltage icon 'V' and the component or parts of the circuit across which the voltage is to be measured. This instantiates the variable arrays, Use, with, for example, 1 which accesses the voltage table and Comp, which holds the coordinates used to map between the terminals and their locations in the table of

voltages. Like the Voltage specialist, the Current and Resistance specialists calculate their values in a similar manner.

5.3.2 Answering factual questions

Factual questions in the form of component specifications can also be posed to Circuit II. By selecting the desired component with the appropriate mouse button the variables x,y and button% are instantiated with values which are then used to invoke the procedure Spec_Check, for example, to give information about the beta of a transistor.

5.3.3 Inserting faults

As well as being able to take various measurements across different parts of the circuit, faults may also be inserted. The specialist InsertFault is invoked when the array Fault is instantiated with a numeric value associated with that fault (for example, short = 5) and Comp, the component array, is instantiated with the x and y coordinates of the component to be faulted. A new file containing the modified circuit configuration is written and is used as input to the circuit simulator to compute new tables which reflect the modifications. In this new context, measurement specialists will reference these tables when invoked.

5.3.4 Replacing parts

When a random fault is inserted into the circuit students can also ask that a part be replaced. When this happens the specialist Replace is invoked which asks how the particular component to be replaced is faulted (for example, open, short, high, low, etc). If the student cannot correctly identify the faulted component or the manner in which it is faulted then the component will not be replaced.

The Replace specialist is a limited implementation of that modelled by SOPHIE. Rather than modelling all possible component faults, when a random fault is inserted a 30 ohm tracking fault to ground is simulated. Unlike SOPHIE, Circuit II does not currently store students' knowledge about the fault nor does it store information about the different ways individual components can be faulted.

By recording knowledge about what SOPHIE thought that a user knew about circuit faults, it was able to judge when to query a request made to replace a faulted component. For example, if a student received an affirmative answer to the question "Is the base-emitter of TR2 open?" then she would know that it was faulted. When requested to replaced it, SOPHIE would use this knowledge and thus not question its replacement. Circuit II currently does not support this feature.

5.3.5 Measurement checking specialist

In either a hypothetical or random fault context a student may wish to know if the measurement just taken is correct or not with respect to a normal working circuit. If the measurement is not correct then that measurement will be a symptom of the fault. SOPHIE allowed students to make this comparison by typing the statement "Is that right?". When this statement was parsed the history list of previous exchanges between SOPHIE and the student was searched backwards to find a meaning for the anaphoric reference "that". Once a meaning was established the circuit simulator SPICE was invoked to determine the correct voltage for this measurement in a fault free context. In Circuit II, when iconic input equivalent to the natural language statement "Is that right?" is received, the procedure SupportMeasurement is invoked and the table of reference (for example,

Volts) and the component's reference in the table for the faulted and fault free contexts are compared and contrasted. The outcome of the evaluation is then reported to the student.

5.3.6 Hypothesis testing specialist

Another SOPHIE feature modelled by Circuit II is the hypothesis testing specialist. When troubleshooting the circuit students will develop hypotheses about the nature of the fault based on the measurements taken and will wish to check if they are correct or not. To determine the consistency of a student's hypothesis Circuit II maintains a history list of all measurements taken prior to the hypothesis being posed. When a hypothesis is presented Circuit II first saves the current context before invoking the circuit simulator under the proposed fault. It then compares all the observed measurements (stored in the history list), one at a time, against those in the hypothesized fault mode. If they are all the same then the student's hypothesis is consistent with the measurements taken thus far, and if not these are reported to the student. As well as reporting differences between an observed and hypothesized measurement it is also compared to what it would be in a normal working circuit. Once the comparison is completed the contents of the history list is erased to make ready for the next set of measurements, and then the previously stored context is resumed. In a similar manner to SOPHIE, Circuit II has the ability to highlight errors in a student's logical way of thinking. A current limitation of the hypothesis testing specialist is its inability to report components which may blow as a fault propagates through the circuit.

5.3.7 Conditional specialist

Students wishing to pose "If... then" types of questions can do so by invoking the conditional specialist which permits graphical equivalents of questions, for example, "If the base-emitter of Q3 opens what will happen to the current through R3?" to be posed. When invoked the conditional specialist saves the current fault-free context and then modifies and writes a new circuit file which reflects the nature of the hypothetical fault. This acts as input to the circuit simulator which computes new values based on the fault and stores them in the table of values for this new context. When measurements are taken these tables are referenced and the student is told of the consequences of this fault.

5.3.8 Explanation specialist

When troubleshooting on the electronic circuit it is possible that a student will be unable to isolate the fault. In this situation the student can evoke the explanation specialist which graphically displays a linear sequence of windows which show measurements which could have been taken to locate it. Within each window, readings for each measurement in the faulted and fault free contexts are given to help students develop their logical thinking skills. In contrast, when SOPHIE users could not find a fault they could ask for help by typing "What could be wrong?". This invoked a hypothesis generation facility which would generate a set of hypotheses which would account for the measurements taken up to that point. Circuit II does not support this facility which is a limitation of the current implementation.

5.4 Simulation Techniques

This section describes the general purpose simulation system used by Circuit II to compute numerical values which are used by the various specialists when answering students' questions. The general purpose circuit simulator, MITEYSPICE (McCabe, Childs, and Clarke, 1988), is an electronic circuit simulator program for the Acorn Archimedes microcomputer. It can perform D.C., small signal A.C., and noise analyses. The mathematical techniques used in the program, and the input syntax are based largely on the Fortran program SPICE (Simulation Program with Integrated Circuit Emphasis) (Nagel, 1971; Nagel et al., 1973) developed at the University of Southern California.

5.4.1 D.C. Analysis package

Analysis of a circuit file within MITEYSPICE involves three stages of analysis. The first step is to create a file of expressions using the Edit facility which describes the circuit's configuration. Next, the file is read by the procedure Reedin which passes the input to Parse which checks that the circuit configuration is a valid one. Next, the parsed input is passed to the procedure Setup which evaluates the file of expressions and allocates space in memory for the array which holds information about each component in the circuit and its relationship to others. The D.C. working conditions are then found by the procedure Analysis and transistor conditions, if present, are found by calling the procedure Iterate since they involve non-linear functions. Data representing the working conditions of the circuit is used to update the voltage and current reference tables used by Circuit II.

MITEYSPICE performs five different types of circuit analysis: D.C. operating point, Sweep, A.C. analysis, Sweep...At, and Noise. The simplest

analysis type is the D.C. operating point which finds the steady state voltages and currents in a circuit. Sweep, by contrast, calculates the values for several operating points to find how one or more component values or other circuit parameters affect the steady-state voltages and currents. A.C. analysis calculates the frequency response of a circuit over a wide range of frequencies. The Sweep...At is similar to the Sweep analysis except that in addition to D.C. operating point calculation at each sweep point, an A.C. analysis at a spot frequency is performed. This allows changes in A.C. response with component values, transistor parameters, etc, to be calculated. The Noise analysis evaluates the thermal, shot, and flicker noise contributions from all the circuit components, and sums them at a defined output node, taking account of the gain between the noise and source and the output point. Thus the total output noise is evaluated. An input point is also defined, and the referred equivalent input noise calculated. The current implementation of Circuit II only uses the D.C. operating point analysis at this present time.

5.4.2 Modifications made to MITEYSPICE

A number of modifications were made to MITEYSPICE so that it could be run automatically from within Circuit II's graphical environment. All program code associated with file editing was removed and replaced by routines which would update an existing file automatically, for example, to reflect circuit modifications made by a student. Instead of writing output data files to disc, the code was modified so that the data was read directly into arrays held in memory.

5.4.3 Introducing faults into MITEYSPICE

Circuit II uses two files which contain identical descriptions of the circuit, one being used for the calculation of normal working values and the other for circuit modifications made by a student. Modifications to the circuit can be made very quickly because the file descriptions are held in memory which eliminate the need to access and save files held on disc. When the circuit simulator is invoked the dynamic arrays holding the reference tables are updated.

5.4.4 Performance of MITEYSPICE on D.C. analysis

Because of the way in which MITEYSPICE uses memory and variables it cannot reside in memory at the same time as Circuit II's program modules. This means that every time a fault is inserted MITEYSPICE has to be loaded in from the ram disc before a calculation can be made. Using this unsatisfactory procedure, each calculation made takes on average 7 cpu seconds.

5.5 Endowing Circuit II with some intelligence

Circuit II exhibits intelligent behaviour through its use of simulation techniques and device specialists when answering certain classes of questions, i.e., measurements and the evaluation of hypotheses. Answers to such questions when posed in a faulted context are computed and compared against their counterpart for a normal working circuit to decide upon the validity of the question or hypothesis.

5.5.1 Inference generation by simulation

The circuit simulator, when invoked, enables the circuit model containing all the knowledge about how the device is connected to be simulated. By altering the model a hypothesis of the form "If x then y" can be modelled, for example, "If the base of Q1 were open then the base-emitter voltage of Q3 would be low". In this example "the base of Q1 were open" is proposition x and "the base-emitter voltage of Q3 would be low" proposition y which describes the symptoms of such a fault. By modifying the circuit model so that the base of Q1 is open and then running the simulation, the validity of proposition y can be determined by checking it against the computed value.

This facility gives Circuit II, like SOPHIE, the ability to respond intelligently to hypothetical questions posed by students. However, unlike SOPHIE, posing hypotheses of the form "If x then y" also has the effect of automatically suspending the normal working context and creating a newly faulted one based on the posed hypothesis. In SOPHIE, a context switch had to be made using the SAVE command, saving the current context before a user's hypothesis could be explored.

The act of posing a hypothetical question, the automatic digression from one context to another, the presentation of an intelligent response from the system, it is suggested, matches and extends the "friendliness" and usability of Circuit II beyond that of SOPHIE by making it more conversational.

5.5.2 Hypothesis evaluation (testing)

In troubleshooting the circuit in a random fault context users will generate hypotheses about the nature of the fault and wish to test them out. Circuit II is programmed to interpret user mouse selections made in this context, for

example, "Is it possible that the base of Q2 is open?" as posing a hypothesis about the nature of the fault. To pose this hypothesis a user would select, in either sequence, the icon representing an open-circuit fault and the base of Q2 using the second mouse button. By contrast, in a hypothetical fault context the same actions would constitute a modification request, for example, "Open the base of Q2". The interpretation of mouse selections in this way makes Circuit II seem intelligent about topic shifts and extends the range of questions which it can handle using a limited iconic set.

When their hypothesis is presented to Circuit II the circuit model is modified to reflect the nature of the hypothesized fault and then evaluated by invoking the simulation process. All measurements taken prior to the hypothesis being presented are repeated under this hypothetical model and compared against one another by an evaluation specialist. If the measurements are not consistent with those taken under the hypothetical model then the student's hypothesis about the nature of the fault is incorrect.

5.6 Summary

This chapter has given a comprehensive overview of Circuit II's capabilities and has shown that its graphical interface, using a limited set of icons, can model a wide range of SOPHIE dialogue within different contexts.

Circuit II's use of free-order syntax, like Circuit I, supports analogues of surface linguistic features anaphora and ellipsis by allowing commands and arguments to be modified. Combinations of these commands and arguments allow measurements to be made, and in conjunction with menus allows components to be modified. This form of syntax has been extended in Circuit II to support an additional command-argument structure which

allows components to be modified directly thus removing the need for menus. Depending upon the context, specifying combinations of these command-argument structures, in any order, permits user-expressions of the form "If x then y" and "Could x be wrong".

Circuit I, like most direct manipulation interfaces, supports a limited form of deixis where pointing with the mouse means only one thing. Circuit II extends this by supporting a more general form of deixis. Using the three mouse buttons available, pointing can mean a) btn. 1: a request for a measurement, b) btn. 2: a modification request, or c) btn. 3: a component specification request.

Rather than using explicit commands like SOPHIE to effect a topic shift, Circuit II avoids their use by re-implementing Reichman's notion of natural language cue phrases in the direct manipulation style. Within a normal working circuit context, selecting a command representing a fault or an argument representing a component with the second mouse button acts as a cue phrase (for example, "by the way") which effects an automatic topic shift. It is suggested that this provides a smoother way of effecting a transition than the explicit commands used by SOPHIE.

Like SOPHIE, Circuit II only models one circuit, an audio video amplifier circuit. This circuit was chosen because it was reasonably complex and thus would enable a wide range of SOPHIE questions to be modelled.

Circuit II is not a full implementation of SOPHIE and thus has limitations on what it can do. Since Circuit II has not been designed as an instructional system, but rather to demonstrate the practicality of modelling natural language graphically, these limitations have not affected the outcome of this

research. In the following chapter, the design of Circuit II's interface is discussed.

Circuit II - Tracking intentions within & between contexts

6.1 Introduction

In Chapter 5 the detailed design of Circuit II and its constituent parts were discussed. In this chapter the interface techniques which Circuit II uses to model more complex SOPHIE dialogue are presented.

Circuit II, as already explained in §5.2, uses a graphical representation of an electronic circuit through which users pose questions and make requests. By contrast, in SOPHIE, all such measurement and modification requests were made through its natural language interface. The use of a graphical representation of an electronic circuit for communication purposes represents a major difference between Circuit II and SOPHIE.

Circuit II draws upon the work of SOPHIE and Reichman's (1981,1986) work on conversational coherency. One objective in implementing Circuit II was to expand on the work of Circuit I (Singer, 1990), as discussed in §3.0, which established that small portions of SOPHIE dialogue could be modelled graphically. Unlike SOPHIE, Circuit II provides users with a direct manipulation environment within which they interact with an audio video amplifier circuit. For example, users may take voltage, current and resistance measurements of circuit components. A second objective was to explore and establish the place of Reichman's work on *cue phrases* within Circuit II's graphical environment to support topic shifts between contexts. For example, SOPHIE allowed users to switch between contexts to explore a hypothesis of the "If x then y" kind. To achieve this, it used the explicit commands: "SAVE" to save the current context, and "RESTORE" to return to it once the hypothesis had been explored. Circuit II draws upon Reichman's work on conversational coherency who has suggested that conversational participants use specific cue phrases to signal different types

of conversational digressions. A third objective in implementing Circuit II was to determine if its support of free-order syntax would offer any significant advantage over other direct manipulation interfaces currently used.

Modelling the same range of dialogue as SOPHIE I has meant re-implementing many of its designed features as discussed in Chapter 5. These features, incorporated into Circuit II, include procedures for handling simulation processes, measurements, faults, replacing parts and hypothesis testing. In response to the many questions which can be posed graphically to Circuit II, all answers are given in manipulated canned text.

6.2. Developing a more flexible syntax

The syntax of Circuit I described in §3.2 and §3.4 showed how simple command-argument combinations made by users allowed them to pose questions of the form "What..." and "If x then y". In addition, by varying the command or argument portions of the syntax mouse selections could be made which performed a similar role to natural language anaphora and ellipsis. Circuit I was a simple prototype model and only intended to demonstrate the feasibility of modelling a small sample of SOPHIE dialogue.

Circuit II is a fairly full re-implementation of SOPHIE which extends the work of Circuit I by graphically modelling most of the dialogue handled by SOPHIE I. To do this has meant developing a graphical interface of a large circuit of comparable complexity to SOPHIE's Heathkit IP-28 regulated power supply. Using this as a platform, graphical techniques have been developed to extend the flexibility of the command-argument syntax by supporting a larger range of commands and arguments. Commands representing specified uses (for example, voltage, short-circuit) are supported as well as arguments representing components (faulted and un-

faulted). More complex types of questions can also be posed containing two arguments and two commands, where each argument or command can be modified in favour of another. Unlike Circuit I, Circuit II supports three different contexts representing a normal, hypothetical and faulted circuit which users can move between. All user input is made through a direct manipulation environment with all system responses being made in "canned" text.

6.2.1 Circuit II syntax

Work on Circuit II has sought to provide users with an interface which supports either a command-argument or argument-command syntax. Currently, Circuit II supports two different forms of command-argument and argument-command syntax, which may be used either to ask for measurements or to insert faults. The information about which is active is supplied by the user selecting one of two mouse buttons.

The first form of command-argument and argument-command syntax supported by Circuit II which allows measurements to be taken is shown in figures 6.1a and 6.1b. The resistors R1, R2 and R3 represent arguments whilst V (voltage), I (current) and R (resistance) represent commands. When, for example, an argument is supplied prior to a command, as in statement 1 of figure 6.1a, the component represented by that argument will display a question-mark. This feature has been incorporated into Circuit II as a means of indicating to a user that some further selection, for example, a voltage (V), current (I) or resistance (R) measurement is required before a meaningful response can be given.

Circuit II differentiates between certain argument-command expressions which reflect the different types of measurements being taken. In statements 2 and 4 of figure 6.1a, where voltage (V) and resistance (R) are being measured, only the inspection nodes 'across' a component are

activated. Voltage and resistance measurements are only meaningful when taken between two points. This is in contrast to statement 3 of figure 6.1a, where the body of the component is activated since current flow (I) can only be measured 'through' it.

Figure 6.1b demonstrates how command-argument expressions may be used, for example, to take voltage measurements of more than one component. When a command is activated first, prior to an argument, as in statement 1 of figure 6.1b, the mouse pointer changes colour to reflect the chosen command and then waits until a component or inspection point is selected. The pointer will turn red when voltage is selected, blue for current and grey for resistance.

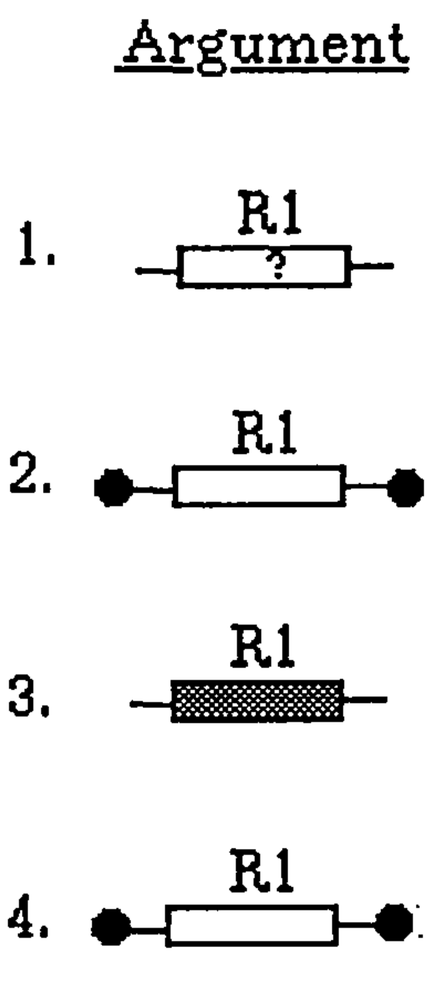


Figure 6.1a argument-command expressions

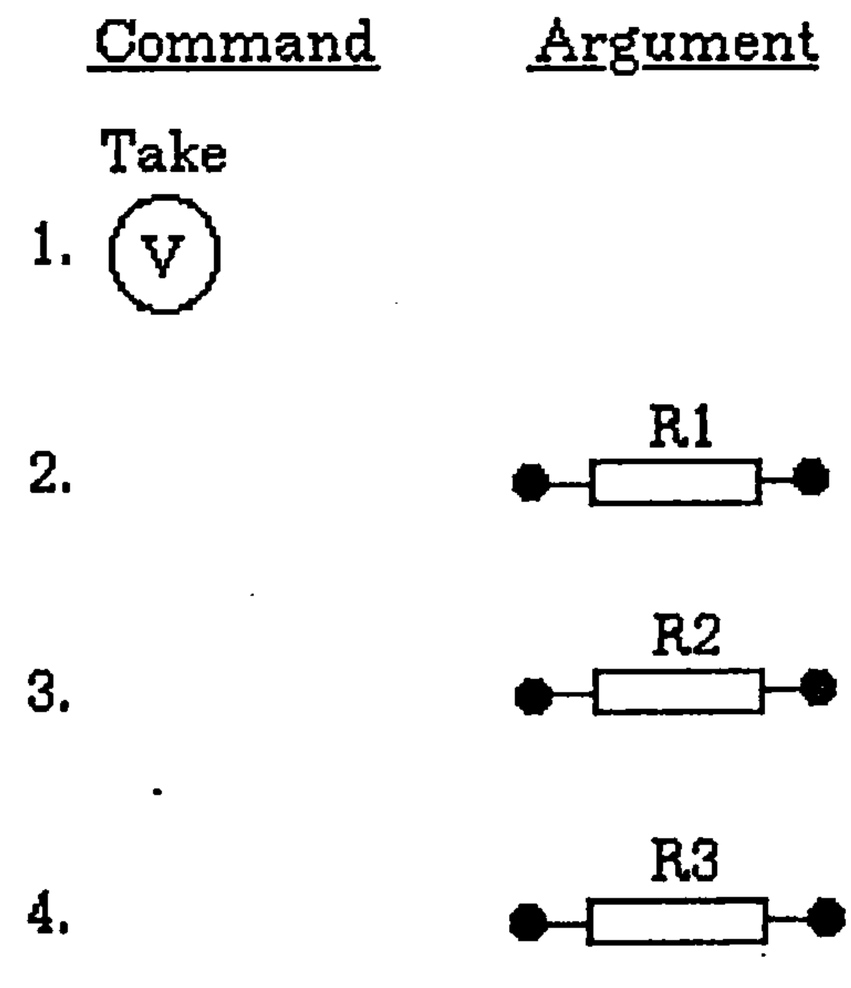


Figure 6.1b command-argument expressions

The second form of argument-command and command-argument syntax supported by Circuit II, which allows components to be faulted, is shown in figures 6.2a and 6.2b. The arguments - Q1, Q2, R2 and R5 are transistors and resistors, and commands - 'short-circuit' and 'open-circuit' are the different faults which can be applied to them.

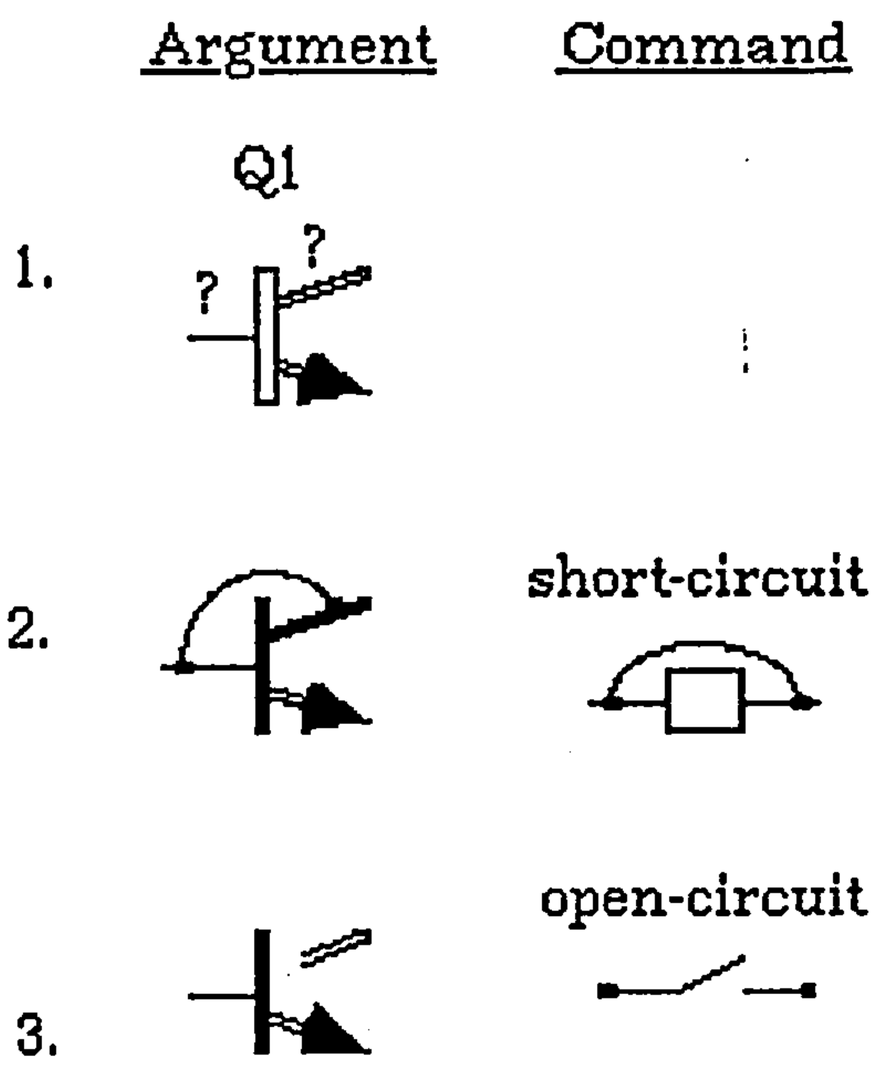


Figure 6.2a argument-command expressions

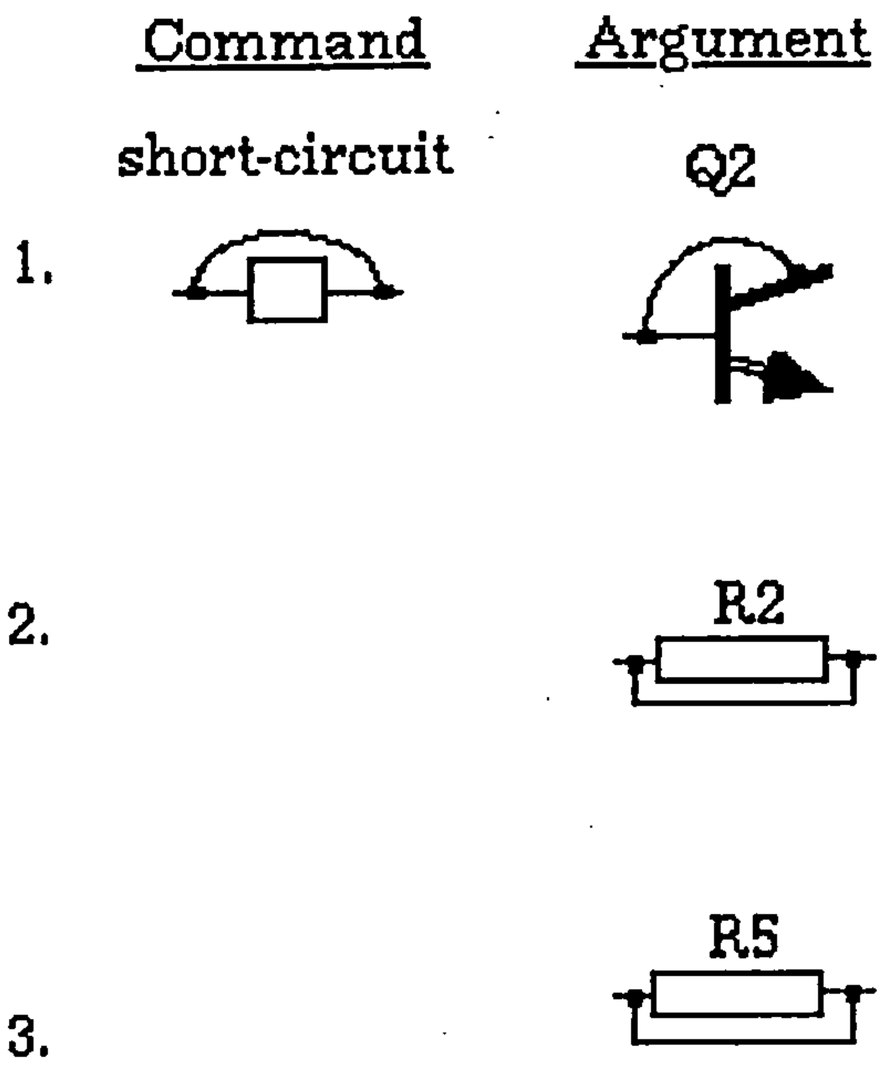


Figure 6.2b command-argument expressions

As previously described, arguments like resistors, specified prior to a command display a question-mark. Transistors however may display more than one question-mark depending upon how they are activated. Statement 1 of figure 6.2a indicates that Q1's base-collector has been activated and is waiting for some command action to be specified. The command 'short-circuit' has been specified in statement 2 and the base-collector of Q1 has been shorted. Circuit II remembers which parts of a component have been faulted so that when another command is specified, for example, 'open-circuit' in statement 3 of figure 6.2a, it responds appropriately by opening the base-collector of Q1. Commands representing different faults can also be applied to different components as shown in statements 1-3 of figure 6.2b.

6.3 Linguistic analogy

Another consideration in providing a less rigid form of syntax was to demonstrate that graphical interfaces are capable of providing some of the functionality of natural language. In particular, an attempt has been made

to demonstrate that anaphora, ellipsis and deixis, which are important linguistic mechanisms in natural language, have their counterparts in a graphical interface.

6.3.1 Varying commands

"Anaphora reference is one way of making the identity between what is being expressed, for example, ['it'], and what has been expressed, for example, ['R1']" (Crystal , 1980). That is, 'it' is an anaphor pointing back to a previously used word. To derive the intended meaning of an utterance a speaker must search the previous dialogue in order to decide which word the anaphor refers to. For example, statements 3 and 5 of excerpt 1 illustrate how the engineer E1 anaphorically refers to R1, a previously mentioned object.

Excerpt 1

E1. 1. What is the voltage across resistor R1?

E2. 2. It's 1.667 volts.

E1. 3. What's its current?

E2. 4. 0.78 milliamps.

E1. 5. How about its resistance?

E2. 6. It's about 22 ohms.

The argument-command expressions of statements 2 - 4 of figure 6.1a do permit users' mouse selections to be viewed as having the same role as those anaphoric references of statements 3 and 5 in Excerpt 1. However, these mouse selections are *not* anaphoric in the natural language sense because there is no possibility of resolving them incorrectly .

6.3.2 Varying arguments

Ellipsis on the other hand may be thought of as "a sentence where, for

reasons of economy, emphasis or style, a part of the structure has been omitted, which is recoverable from a scrutiny of the context" (Crystal, 1980). Sometimes it is unclear what an elliptical construction refers to, as illustrated in the dialogue between two electrical engineers E1 and E2 in statement 3 of Excerpt 2.

Excerpt 2

E1. 1. What is the voltage across the inspection points P1 and P2?

E2. 2. It's 5.5 volts.

E1. 3. P4?

E2. 4. Between P2 and P4 it's 7.6 volts.

To resolve the ambiguous utterance of statement 3 in Excerpt 2, E2 has to infer what E1 wants. The type of inferencing E2 will apply could depend upon many things. For example, in statement 4 of Excerpt 2, perhaps E2 decided to give E1 the most meaningful measurement of the possible alternatives, although he could have asked him "Do you mean between P4 and P2?". Circuit II attempts to provide a close approximation to natural language ellipsis by attempting to infer what is meant when an ambiguous situation involving inspection points occurs. Statements 1 and 2 of figure 6.3 shows how statements 1 and 3 of excerpt 2 would be modelled graphically in Circuit II.

Currently, Circuit II has only one mechanism which attempts to resolve ambiguous mouse selections which involve inspection points. Given that no previous inspection points have been highlighted prior to P1 and P2 of statement 1 in figure 6.3, a default rule is applied which dehighlights the first activated inspection point, P1, in favour of the last, P4. Whilst this is a simple rule, it has at least a fifty per cent chance of guessing what the user intended.

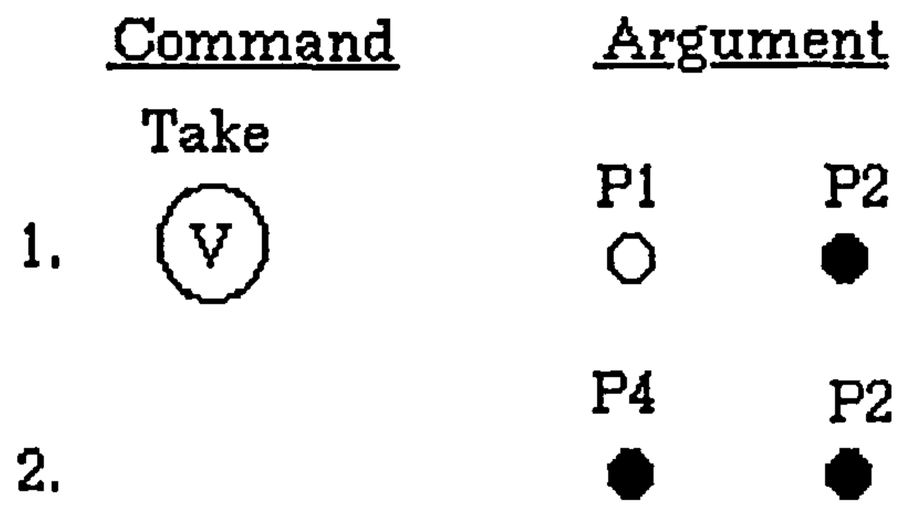


Figure 6.3 Ellipsis resolution

6.3.3 Deixis

Our ability as speakers to point (for example, at an object) to disambiguate it from others, or to refer to its different parts is a "culturally established method of identification" (Lyons, 1981) called 'deixis'. Of the three different kinds of deixis which can be expressed within the "socio-spatio-temporal axes of the ordinary speech situation" (Jarvella, 1982), here, we are only concerned with spatial deixis such as 'here' and 'there'. For example, a resistor in an electronic circuit could be distinguished from others by pointing to it when speaking, as E1 does in statement 1 of Excerpt 3. In statement 3 of excerpt 3, when E1 points to it again he means something entirely different, i.e., he wants its specification.

Excerpt 3

E1. 1. What's the voltage across this resistor *here*?

E2. 2. Across there it's 8.34 volts.

E1. 3. OK. what's *its* specification?

Our ability to express different meanings when we point to an object or its parts is an important feature of spatial deixis. This more general form of deixis has been modelled within Circuit II in an attempt to enhance the 'naturalness' of the direct manipulation environment for the user. As discussed in §1.0, supporting deixis within a direct manipulation interface

is not new. However, many such interfaces have a rigid convention such that pointing, only means one thing, which is inflexible. For example, pointing often only means icons, so to point to a window you can't point anywhere in the window but only to a special tiny area, such as the title bar. Circuit II allows this form of deixis to be modified depending upon which mouse button is pressed, and so is more like the general form of deixis found in natural language than in some interfaces.

An example of spatial deixis in Circuit II is given in statement 2 of figure 6.4.

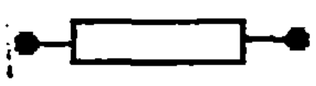
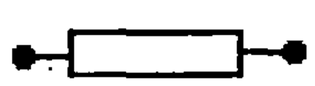
	Command	Argument	Btn.	Result
1.	Take 	R2 	1	The voltage across R2 is 7.5290 volts.
2.		R2 	3	The specification of R2 are: Carbon Film resistance -- 680 ohms +/- 5%. Power/Rating -- 0.25 watts. Anyway, as I was saying across here it's 7.5290 volts.

Figure 6.4 Spatial deixis

Statement 1 shows how a user can, by clicking button 'one' of the mouse over the voltage and resistor icons request a voltage measurement across that component. In statement 2 however, clicking button 'three' of the mouse over the same resistor displays its specification. Given that a previous measurement has been taken in statement 1, when a component's specification is requested Circuit II assumes that a brief conversational digression has occurred. The specification is displayed for a set time before being replaced by the cue phrase 'Anyway...' to signal the resumption of the previously interrupted context. No cue phrase is given if a measurement

request across or through any component has not been previously requested. In this situation no digression has occurred. The chosen convention within Circuit II is to highlight the body of a component or its parts when current, voltage and resistance is being measured. A specification request refers to the whole component, not just a part of it.

6.4 Textual output

A design consideration during the implementation stages of Circuit II was how it should respond to users' graphical input. Like SOPHIE, natural language output was chosen because of the need to support and make explicit CircuitII's use of linguistic analogy as described in section 6.3 above.

6.4.1 Argument variation responses

Although Circuit II responds in 'canned' text, its output has been designed to echo mouse selections analogous to anaphora, ellipsis, deixis and topic shifts using a gradation of responses appropriate to the situation. Statements 2 and 3 in figure 6.5 give an example of how Circuit II grades its responses when mouse selections are elliptical.

6.4.2 Augument-command variation responses

Combining the two forms of argument-command and command-argument syntax supported by Circuit II enable it to handle questions of the "What.. If.." form. Statement 1 of figure 6.6 shows how Circuit II can handle mouse-selections which are analogous to a typical user question posed to SOPHIE "If the base-collector of Q1 shorts what will the voltage across the base-collector of Q3 be?". Statements 2 and 3 of figure 6.6 show typical responses which Circuit II gives when the command or argument part of a question is varied by a user.

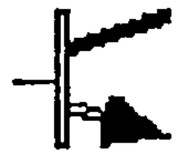
	Argument	Command	System Response
1.	R1 	Take (I)	The current through R1 is 0.221 amps.
2.	Q1 		Through Q1's collector its 0.0156 amps.
3.	R3 		.61 amps.

Figure 6.5 Graded responses for argument variations

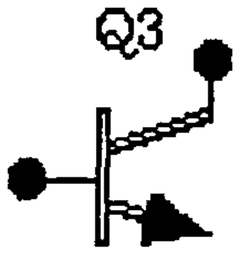
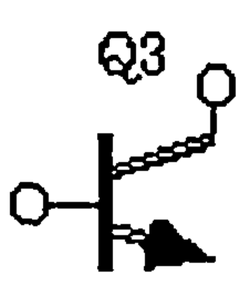
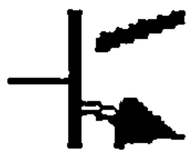
	Command	Argument	Command	Argument	System Response
1.	Short-Circuit 	Q1 	Take (V)	Q3 	If the base-collector of Q1 is shorted then the voltage across the base-collector of Q3 will be 4.1901 volts.
2.			Take (I)	Q3 	If the base-collector of Q1 is shorted then the current through the base of Q3 will be 0.0207 milliamps.
3.	Open-Circuit 	Q1 			If the emitter of Q1 is open then the current through the base of Q3 will be 0.0207 milliamps.

Figure 6.6 Different responses for argument-command variations

6.4.3 Challenge-support responses

Like SOPHIE, Circuit II permits users to check if a measurement, given under faulted conditions, is correct or not with respect to a normal working circuit. To check such a measurement in SOPHIE, users typed "Is that correct?" or "What should it be?". Whilst Circuit II's mechanism for allowing users to ask such questions is different from SOPHIE's, it has been designed to achieve the same result. If a user asks Circuit II a question in a

faulted context, as in statement 1 of figure 6.7, the system responds with the appropriate answer. To check if this answer is correct or not with respect to a normal working value, the user clicks on the tick ('√') which has the effect of changing the mouse pointer to a '√'. When the pointer is placed over the systems answer and the mouse button is pressed, the challenge-support window is displayed as illustrated in statement 2 of figure 6.7.

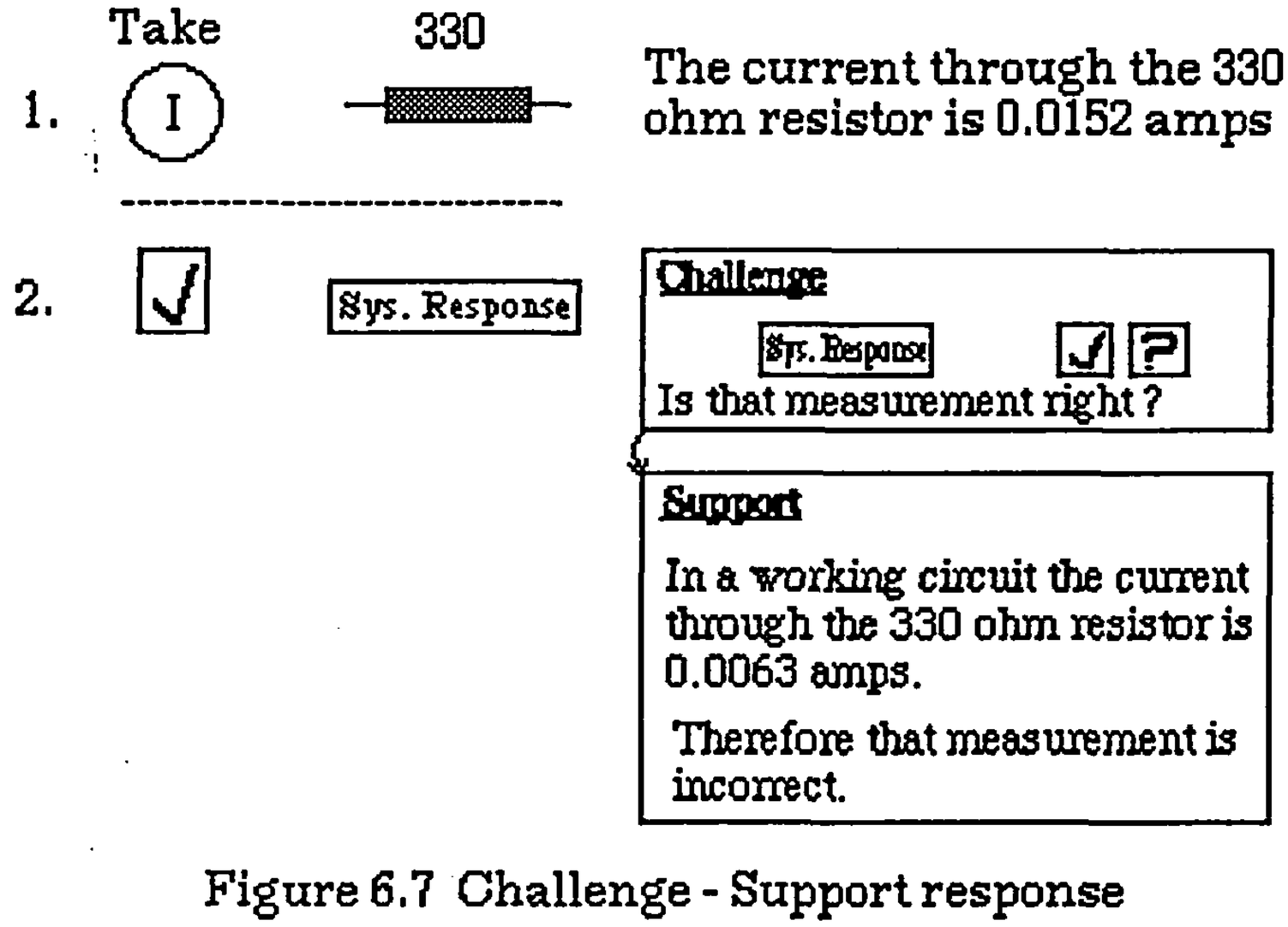


Figure 6.7 Challenge - Support response

The windows have been labelled challenge-support to signify a user's request for an elaboration (challenge) of a response given by the system and its answer (support). The reason for adopting these terms was to establish if any users saw the activation of this mechanism as directly challenging a response given by the system or as a request for more information. When the challenge-support mechanism is activated a window labelled 'challenge' displays the icons representing a user's graphical action and their natural language meaning. This window is shown linked to the 'support' window within which a SOPHIE like response is given which is Circuit II's way of supporting the previously given answer.

6.4.4 Replacement responses

Within a random inserted fault context a user is free to troubleshoot the circuit in order to locate it. When users think that they have located the faulty component they may ask the system to replace it. Typing, for example, REPLACE Q3 within the SOPHIE environment invokes the REPLACE specialist which questions the user about how the component is faulted. If the user either correctly identifies the fault or any of the faulted parts then the specialist would accept this and describe the complete fault mode before replacing the component. Failure to do this would prevent the component from being replaced.

Invoking the REPLACE specialist in Circuit II is done graphically by clicking upon a soldering-iron icon which causes three things to happen. The icon becomes inverted to indicate that it is now active and adjacent to it another box appears as shown in statement 1 of figure 6.8. The mouse pointer also changes from an arrow to the word 'CHANGE', (see statement 2) which, if it is activated over a component displays that component in the box adjacent to the soldering iron (see statement 3). The system responds by asking how the selected component is faulted as shown in statement 4. Selecting the appropriate icon which represents the suspected fault type (the command short-circuit) causes the mouse pointer to change reflecting this as shown in statement 5. By applying the mouse pointer to the suspect parts of the component (the argument) the user can indicate to the system what they think is wrong with it. If, like SOPHIE, Circuit II finds their answer incorrect, it responds with the appropriate message as in statement 6.

6.4.5 Hypothesis responses

Circuit II provides users with a means of graphically posing their hypothesis of what they think is wrong with a faulted circuit and having it critiqued by the system. Circuit II's hypothesis critiquing specialist, like

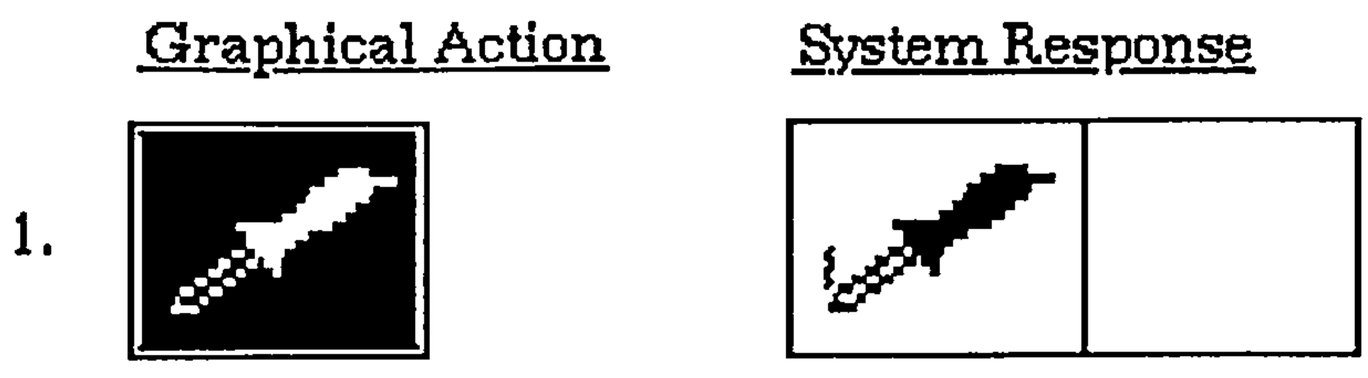


Figure 6.8 Replacement responses

SOPHIE's, determines if the symptoms of the fault agree with the symptoms observed by the user. If all of the measurements observed by the user are consistent with those of the suggested fault, then the proposed hypothesis is consistent. Any measurement which conflicts with the proposed hypothesis is highlighted to the user as illustrated in figure 6.9.

6.5 Conversational coherency

The use of free-order syntax in Circuit II, as described in section 6.2, and the use of linguistic analogy, as described in section 6.3, it is suggested, permits users' mouse selections to be interpreted in a way which may well mirror their meaning more closely as they occur at the cognitive level during a natural language exchange. The evaluation of Circuit I (Singer, 1990)

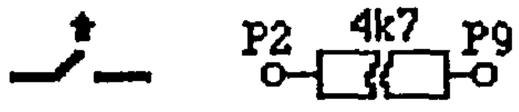
Graphical Action	Sys. Response
	You have suggested that the 4k7 ohm resistor could be open. Hmm... let me think about that for a moment. That fault conflicts with the following measurements: You observed that the voltage across P0 and P3 was 2.4338 volts. *If the 4k7 ohm resistor were* open it would be 7.4680 volts. In a working circuit the voltage across P0 and P3 is 7.4680 volts. There are no further measurements which conflict with your hypothesis.

Figure 6.9 Hypothesis responses

(discussed in §4.0) and an observational study conducted with Circuit II (Singer, 1992), which is discussed in §7.0, support this hypothesis.

Circuit II attempts to extend the usefulness of free-order syntax expressions, which allow users to ask for measurements or insert faults, by allowing them to be combined. This enables users to make mouse selections which have a role analogous to "If x then y" questions in natural language. Circuit II's ability to handle such combinations greatly extends the range of SOPHIE-like questions it can model. For example, figure 6.10 shows how Circuit II can handle mouse-selections which are analogous to a typical question posed by users to SOPHIE. "If the base-collector of Q1 shorts what will the voltage across the base-collector of Q3 be?" Circuit II's ability to model such combinations permits us to model another important feature of conversation - topic shifts. As outlined in the introductory section, SOPHIE used "SAVE" and "RESTORE" commands to effect a contextual change and

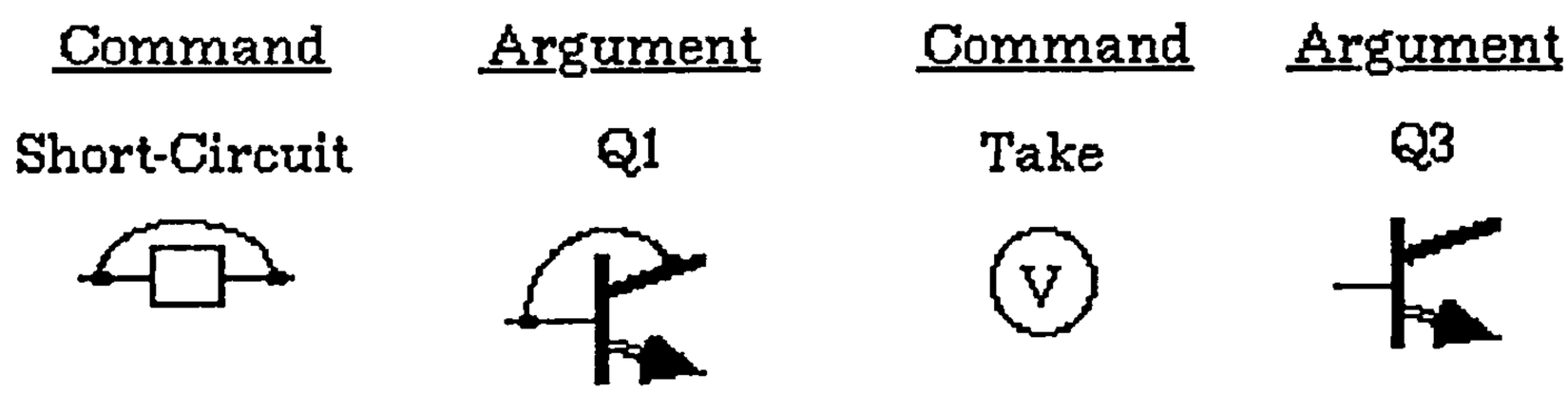


Figure 6.10 Asking a hypothetical "If x then y" question

resumption. This work has sought to develop graphical analogues to these two mechanisms by exploring Reichman's (1981, 1986) work on conversational coherency, in particular her work on how cue phrases are used to signal a conversational digression and return.

6.5.1 Reichman's theory of conversational coherency

Participants in a conversation have certain expectations that each will be able to follow the twists and turns as it proceeds. To do this they share some kind of implicit knowledge of the rules of conversation which they apply in order to fulfill various discourse functions of the other. Such knowledge enables listeners to delineate the boundaries of what Reichman (1985) calls *context space* structures through which the currently relevant discourse context, in terms of the utterances made, are to be interpreted. The use of cue phrases by a speaker indicates that a context space boundary point has been reached and simultaneously indicates the kind of conversational move which is about to take place.

Although a conversation consists of many conversational moves, Circuit II only models the *Interruption* and *Return to previously interrupted context space* (Reichman, 1985). Interruptions often occur during conversations where one participant will shift the topic to a related but tangential subject. This may be signalled by cue phrases such as "Incidentally," "By the way," or "Oh, I forgot to tell you." When this happens the interrupted context (current context space) is temporarily suspended whilst the new topic is

discussed with the expectation that it will be resumed once the interruption is complete. Resumption of the previously interrupted context space may be signalled by words like "Anyway" or "In any case."

6.5.2 Graphical equivalent of cue phrases

Circuit II can model such a conversational digression because an electronic circuit can be in one of two states, namely faulted or unfaulted. Automatic digression to a hypothetical fault context is effected by depressing the second mouse button over a fault (command) or component (argument) icon. Prior to a digression, the current state of the icons in the normal working context are saved before the surrounding window changes colour, from green to red, signifying a faulted context. The icon (argument or command) which has been selected with the second mouse button will highlight at the same time as the window changes colour. It is suggested that such an action is analogous to the cue phrases "Incidentally" or "By the way" which signals a digression in everyday conversation. If an argument icon is highlighted first, using the second mouse button, then a question-mark(s) will appear beside the icon signifying to the user that the system is waiting to be told in what manner the selected component should be faulted. If a command icon is highlighted first then the system will wait until the component, which is to be faulted, is selected. Within a hypothetical fault context, questions may be posed, based on the selected fault, using the different argument-command, command-argument combinations with the first mouse button or changing the nature of the fault with the second. When a user wishes to return to an unfaulted context they do so by selecting, with the first mouse button, a 'label' which is marked 'Norm Circuit'. This action, it is suggested, is analogous to the cue phrases "Anyway" or "In any case" which signals the resumption of a previously suspended context in everyday conversation. Icons representing user questions, posed prior to a digression, are reactivated, the window changes colour from red to green

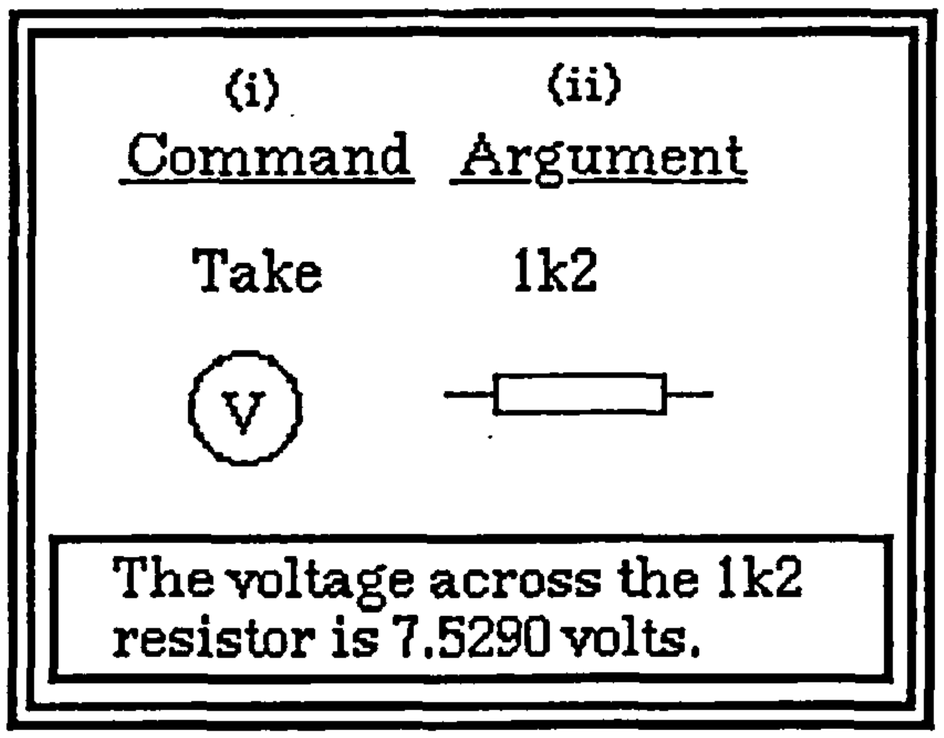
and the system responds with, for example, "Anyway, as I was saying, across here it's 7.5290 volts.". Figure 6.11 illustrates Circuit II's graphical clue word mechanism for handling an interruption-resumption type conversational move.

6.5.3 Capturing meaning through context

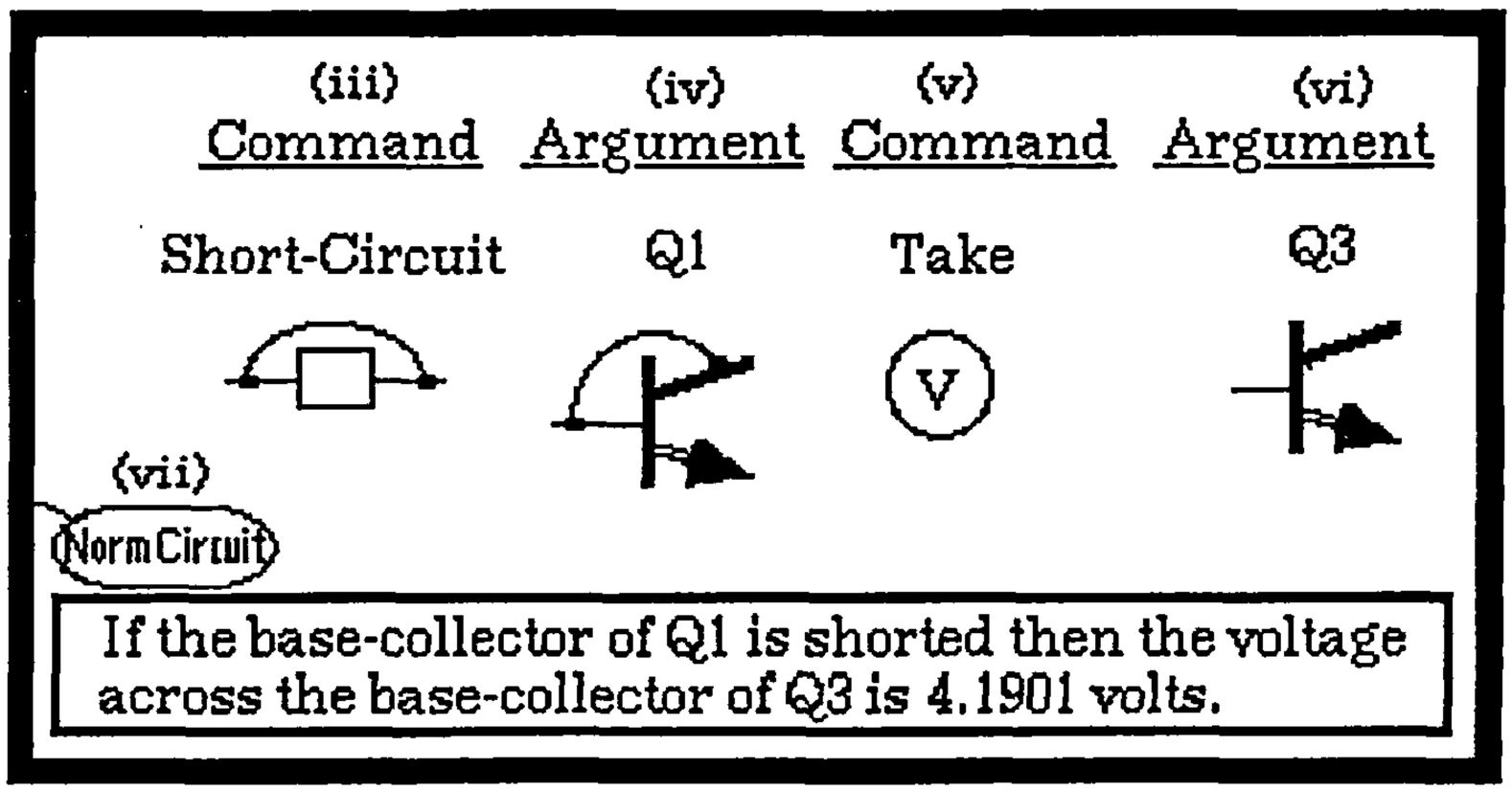
Circuit II, as well as allowing users to digress from a normal working circuit to a faulted one where their hypotheses can be explored, provides users with the means of requesting, to the system, that an unknown fault be inserted. In this scenario, rather than starting with a normal working circuit context, users start with a randomly faulted one. Users must apply their troubleshooting skills in order to locate the fault by taking measurements of its various parts. Like SOPHIE, Circuit II provides two mechanisms which can be activated by a user, to either replace a component if they think they know which component is faulty, or alternatively to pose their hypothesis of what they think is wrong to the system and have it critiqued.

The mechanism for replacing a component, illustrated in figure 6.8, can only be used when a random fault has been inserted into the circuit. This is in contrast to the mechanism handling user posed hypotheses presented within a random fault context, which is also used to fault components within a hypothetical fault context. Figure 6.12 illustrates how the same mouse selections can be interpreted differently through contextualization. It has already been explained how an automatic digression from a normal working circuit to a hypothetical fault context occurs, and how this is signalled to the user visually by means of the surrounding window changing colour from green to red. In this context the user is free to decide which component to fault using either an argument-command or command-argument combination with the second mouse button. In the random fault

(a). Normal working circuit



(b). Digression to a faulted Circuit



(c). Return to a working circuit

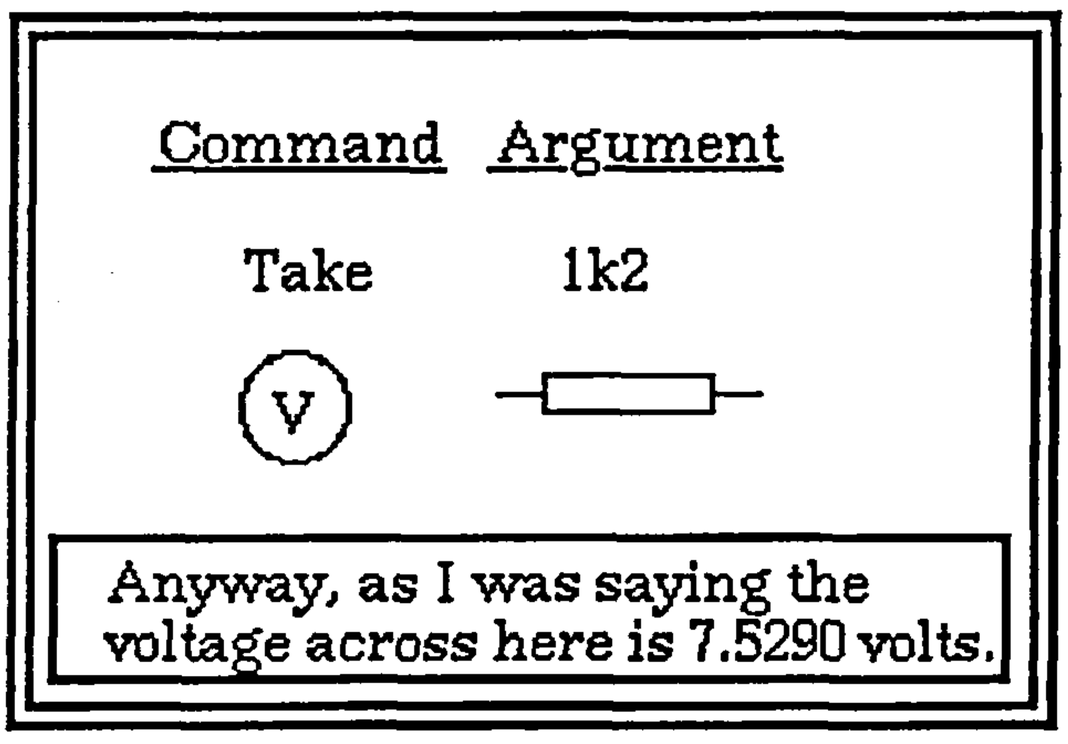


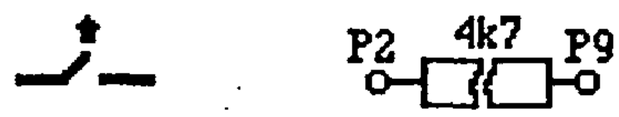
Figure 6.11 Signalling a return to a previously suspended context

context however, entered by clicking on the appropriate icon, the same mouse selections used in the hypothetical fault context are used to convey to the system the users' hypothesis that this could be the fault. Rather than treating the mouse selections as a command to fault a component in a

specific way, Circuit II interprets the selections as a command to *evaluate* the user's hypothesis based solely upon the measurements taken up to the time of their conjecture.

Hypothetical Fault Context

Graphical Action

Command	Argument
open-circuit	

Sys. Response

Now the 4k7 resistor is open.

(Wait a moment... calculating circuit values.)

Random Fault Context

Graphical Action

Command	Argument
open-circuit	

Sys. Response

You have suggested that the 4k7 ohm resistor could be open. Hmm... let me think about that for a moment.

Figure 6.12 Capturing meaning through context

6.6 Summary

The work of Circuit I, which was discussed in §3.0, demonstrated how free-order syntax (command-argument and argument-command) combinations could be supported and varied within a direct manipulation interface. Circuit II has extended this syntax by modelling a greater range of commands and arguments within a more complex circuit. Rather than using command-argument combinations in conjunction with a pull-down menu to supported complex questions of the form "If x then y", Circuit II does so using direct manipulation techniques. These techniques enable most of the complex questions accepted by SOPHIE to be modelled by Circuit II.

Circuit II extends the rigid form of deixis supported by Circuit I and many direct manipulation interfaces. By dedicating each of the three mouse buttons to a specific function, that is, measurement requests, modification

request and specifications requests, allows the form of deictical reference to be modified. This extends the flexibility of the free-order syntax supported by Circuit I and the degree of expressivity available to users. This form of deixis more closely approximates the general form of deixis found in natural language.

Circuit I also showed how mouse selections made in a graphical environment can perform an analogous role to specific natural language features like anaphora and ellipsis. Further analysis of these analogies in Circuit II have revealed that mouse selections made which vary the command portion of the command-argument syntax are not anaphoric because responding to them does not involve searching for a previously mentioned referent. For ellipsis the case is stronger. Inspection points representing arguments can be varied in a similar manner to circuit components. However, unlike components whose selection is unambiguous, Circuit II has to choose which inspection point of the two currently selected to de-highlight in favour of a newly selected one. Whilst it uses only a simple default rule this begins to approximate human ellipsis albeit modestly. It is argued that the combined power of modifiable deixis and the free-order syntax of Circuit II enables it to match the brevity, naturalness and convenience that SOPHIE achieved through its use of natural language anaphora and ellipsis. This claim is supported in part by a) the analysis and comparison of extensive examples from SOPHIE and the programs of Circuit I and II, and b) the generation of canned text which is generated in response to analogues of anaphora and ellipsis expressed graphically.

Circuit II has extended SOPHIE's context transition mechanisms (SAVE and RESTORE) by modelling them graphically. Certain mouse selections made perform the role of Reichman's natural language cue phrases which can be used to effect topic shifts between contexts. In direct manipulation interfaces this correspondence is not well matched leading to defects in

usability. It is argued that supporting topic shifts in way leads to a smoother and more natural form of transition than is currently supported in direct manipulation interfaces.

The work presented in §5.0 and §6.0 has presented Circuit II, a system which supports a number of complex mechanisms. In the next chapter, the results of an observational study conducted with Circuit II are reported and discussed.

Summative Evaluation of Circuit II

7.1 Introduction

Chapter six described Circuit II and in particular how its graphical environment handled anaphora, ellipsis, deixis and topic shifts. Another intention in designing Circuit II was to allow users to express their intentions easily, i.e., there would be a 'natural' or transparent mapping between what they wanted to do and how to achieve it in Circuit II. The next step was to establish whether users did indeed find Circuit II easy to use, and the extent to which the principles described in chapter six worked in practice.

This chapter provides a comprehensive summary of how users reacted to and benefited from their interaction with Circuit II. The observational data was gathered over an intensive period of four days using electronic experts who were experienced in troubleshooting faults on electronic circuits.

7.2 Aims of study

The observational study conducted with Circuit II sought to establish: a) if complex SOPHIE dialogue modelled by Circuit II could be understood by users via the free-order syntax supported; b) if deictical mouse selections were considered a conversational digression; c) the degree of linguistic analogy associated with users' mouse selections and Circuit II's textual output; d) whether a measurement made in a faulted context and compared against the same in a normal context were construed by users as challenging the system's answer; and e) what degree of coherency users felt whilst interacting with Circuit II.

7.3 Subjects used

A total of nine subjects participated in the study who were all drawn from The Open University's Electronic Faculty at Walton Hall, Milton Keynes. All subjects were qualified electronic troubleshooters, each with a mimimum of ten years experience None of the subjects had seen or used the system prior to the study.

7.4 Methodology

The approach adopted in collecting data during the evaluation of Circuit II was to employ particular observational techniques. The three techniques chosen for capturing the data were *direct observation*, *video recording*, and *verbal protocols*. Such techniques have been successfully used in similar research (see Wright and Monk, 1989).

Despite the disadvantage that directly observing a user may alter their performance level (see §4.4.3), this technique was chosen because it was necessary to provide corrective information when difficulties were encountered, both during the tutorial and subsequent troubleshooting session.

During an interactive session with Circuit II a subject will highlight many icons in a variety of different ways. Video recording each session was decided upon so that all of the activities engaged in by a subject, including any problems encountered, could be captured and analyzed at a later date. These recordings form a trace and can be compared against the think-aloud protocols which can be categorized against the linguistic phenomena a subject is thought to be expressing, for example, anaphora, ellipsis, deixis and topic shifts.

A video record of graphical and verbal protocols was also made during each

session, in order to provide as full a record as possible of subjects' interactions with the system. The validity of verbal reports as data have already been discussed in §4.4.3.

Whilst each subject volunteered to participate in our study, the time that they could give was limited to a maximum of two hours. In order to elicit as much information from users in the allocated time, a participative approach was adopted where the observer and subject engaged in frequent verbal exchanges when necessary either to make a point or to request further information. No specific questions were put to users during the interaction since it was thought that this would restrict their range of verbal responses. Every session commenced with a thirty minute training period where each subject was shown how to use the different features of the system. The remaining time was used by a subject in a troubleshooting scenario, set up by the researcher, trying to locate an unknown fault.

The data collected from users falls into two categories, *Quantitative* data and *Qualitative* data. Mouse selections are quantitative data in that they form a history list of each interactive session between the user and Circuit II and can be used as means of measuring a subject's competence with the graphical syntax. This is in contrast to subjects' verbal comments which form the qualitative data. Although such comments are subjective they have been used to gauge the effectiveness of the ideas behind Circuit II's conversational mechanisms rather than the mechanisms themselves.

7.5 Results

This section presents and discusses the data collected during the observational study conducted with Circuit II. In analyzing the collected data five categories were devised and used as the basis of analysis.

7.5.1 Analysis of subject protocols

Subjects' individual protocols were analyzed in terms of the following five categories: free-order syntax, deixis, textual responses, Is that right? challenges, and conversational coherency. The protocols were examined to see what information could be derived from them for each subject under these categories.

- **Subject 1**

Free-order syntax

Being able to switch between current and voltage whilst referring to the same previously highlighted component did not surprise S1.

The first time S1 encountered the "?" displayed next to a component he correctly assumed that the system was waiting for him to specify a command action against the argument.

Deixis

The deixis mechanism was not used by S1 during his session with Circuit II.

Textual responses

As far as the textual output of the system was concerned, S1 would have preferred to see the text abbreviated, but appreciated that a novice would prefer a full display of text. The textual display of the system was only confirming what he already thought based on the graphical selections he had made. He found the graphical display clear.

Is that right?

S1 thought he would need to use the "Is that right?" mechanism for quite a while before he could give an opinion about how he preferred to activate it. His initial thoughts were that he preferred the method which involved the

least mouse movements. Despite the fact that each method of activation involved the same number of mouse movements, during the trial S1 activated the mechanism using the '√ - sys. resp.' syntax. S1 did not comment on whether he viewed his actions as challenging the system's previous response.

Conversational coherency

After digressing from a normal circuit to a faulted one, S1 expected it to return to the previous context. He knew that he was in a different context because he recognized the fact that he was taking a new measurement with the 4k7 resistor shorted. He had previously taken a measurement across this component in the unfaulted context. The textual response displayed upon his return to the previous context merely supported his assumption that he was continuing where he left off. He commented that such a mechanism did give him a sense of continuity, that it did seem to follow what he wanted.

A summary of S1's responses within these categories is given in Table 7.1 below.

Free-order Syn.	Deixis	Text. resp.	Is That Right Mech.	Conv. Coh.
Mouse selections: com.-arg.+com. - 18 com.-arg.+arg. - 31 Cat. of comments: com.-arg.+com. - fav. com.-arg.+arg. - none.	Comments: Not used.	Comments: Abbrev.	Method of act. mech: √-sys. resp. Cat. of comments: None. Usefulness of mech: No comment.	Digression: display was clear. Return: Mech. gave sense of cont.

Table 7.1 Summary of subject 1's protocol

- **Subject 2**

Free-order syntax

S2 correctly guessed that to get the voltage measurement of a component, from which he had previously obtained current, it was necessary only to select the voltage icon. He commented that this was a useful feature. He

also said it was useful to be able to vary a previously highlighted component for another to obtain the voltage reading across it.

When he selected arguments first prior to a command which displayed the "?" next to it he had to be told initially what it meant, there after however he interpreted it correctly.

Deixis

S2 commented that the textual response that he received from the system when performing a deictical action was not always going to be appropriate under all circumstances. He thought it would partly depend upon how many measurements of the same type had been taken prior to the request for a component's specification. If a few measurements of the same type were taken prior to requesting a component's specification then the system's response would be appropriate. However, if different types of measurements (voltage, current and resistance) had been taken then it would not feel as though a brief digression had occurred. S2's comments were mainly specific to the textual response and not to the mechanism itself.

Textual responses

S2 preferred the graphical responses, as opposed to the textual ones, that the system gave. To complement the graphical display he would prefer just to have the values given as he does not normally like to read a lot of text.

Is that right?

The fact that the challenge-support window had an automatic delay did not appeal to S2 because he liked to take his time to read things. No comments were made concerning the preferred method of activating the "Is that right?" mechanism and he did not comment on whether he viewed his actions as challenging the system's previous response.

Conversational coherency

With respect to moving between contexts, S2 noticed that windows were changing colour and presumed that each one represented different modes - faulted or working. He was not familiar enough with the system to give an accurate response to the return message but felt that it would be a reassurance to come back to what you were doing previously. In principle, S2 liked the mechanism because he felt that it overcame one of the current limitations of menu systems which forget what you did previously.

A summary of S2's responses within these categories is given in Table 7.2 below.

Free-order Syn.	Deixis	Text. resp.	Is That Right Mech.	Conv. Coh.
Mouse selections: com.-arg.+com. - 30 com.-arg.+arg. - 73 Cat. of comments: com.-arg.+com. - fav. com.-arg.+arg. - fav.	Comments: Unsure	Comments: Short	Method of act. mech: sys. resp. √ Cat. of comments: General Usefulness of mech: No comment	Digression: No comment Return: fav. response

Table 7.2 Summary of subject 2's protocol

- **Subject 3**

Free-order syntax

Altering the specified use, V to I, to get the current through the same component did not surprise S3, for it seemed the right thing to do. Highlighting different components to get the voltage measurements across them seemed to him a natural way to step through the circuit.

The first time S3 encountered the "?" he correctly assumed that the system was waiting for him to specify a command action against the argument.

Deixis

S3 did not use the deixis mechanism during his session with Circuit II.

Textual responses

The response within the challenge-support window 'that measurement is incorrect' was ambiguous to S3 as he was unclear to what it referred. Whilst S3 thought that the textual responses were messy he generally felt that he knew what was going on.

Is that right?

Concerning the "Is that right?" mechanism, S3 said it was better to say that the voltage was correct or incorrect, but felt that it was more appropriate to say something like "the voltage across this component has not been affected by the fault... or it has." No comments were made concerning the preferred method of activating the mechanism or on whether he viewed his actions as challenging the system's previous response. However, S3 thought that it was a very useful facility to have so that measurements taken in one context could be compared in another.

Conversational coherency

When changing context S3 thought that the window changing colour was a thoughtful way of alerting him to something. S3 did not like the way the textual response, upon returning to a normal working context, was worded. He found it ambiguous because he had spent some time in the hypothetical fault mode and found it difficult to remember what had been done previously in this context. S3 said that he found such a response startling because he did not expect computers to be conversational although he conceded the fact that, upon returning to the normal working circuit, having the context restored and the last question re-displayed was a useful facility. As well as having the previous icons highlighted, S3 said that he would like to see the exact textual response which was present prior to the digression. He felt that such a mechanism did give his actions a sense of continuity.

A summary of S3's responses within these categories is given in Table 7.3

below.

Free-order Syn.	Deixis	Text. resp.	Is That Right Mech.	Conv. Coh.
Mouse selections: com.-arg.+com. - 14 com.-arg.+arg. - 14 Cat. of comments: com.-arg.+com. - fav. com.-arg.+arg. - fav.	Comments: None	Comments: Unclear	Method of act. mech: √ sys. resp. Cat. of comments: fav. Usefulness of mech: fav.	Digression: Unfav. Return: fav.

Table 7.3 Summary of subject 3's protocol

- **Subject 4**

Free-order syntax

S4 thought being able to take the current of a component and then change the selection to take the voltage across it was very nice. Just having to make one selection was better than he imagined. No comments were made concerning the selection of components.

He liked being able to combine commands and arguments and thought it was a "nice" way of asking a hypothetical "If x then y" question. "Some instructions concerning the functions of mouse buttons would be helpful" he thought.

The first time S4 encountered the "?" he correctly assumed that the system was waiting for him to specify a command action against the argument.

Deixis

S4 liked being able to take a component's specification and thought that the textual response was good too, for he felt that anything which drew him back to his original train of thought had to be appropriate. Whilst he was not sure about the correctness of the wording, he felt the idea was sound.

Textual responses

Initially S4 was unsure whether or not the system's response should be abbreviated, but then felt that an abbreviated version of the full text response was best.

Is that right?

S4 thought that being able to compare a measurement in a faulted context against one in a normal context was a good feature. He did not comment on the activation of the mechanism because he only used it once at the end of the study. S4 thought that by using this mechanism he was challenging the correctness of the system's previous measurement. As he said, "Yes, it's quite powerful. How practical it would be in reality I don't know."

Conversational coherency

The system's response upon returning to this previous context was clever. He felt that it gave him a sense of continuity, and it reminded him of why he had started the whole series of events previously. It focussed his brain back on the starting point. He realized that the return message had brought him back to where he started.

A summary of S4's responses within these categories is given in Table 7.4 below.

Free-order Syn.	Deixis	Text. resp.	Is That Right Mech.	Conv. Coh.
Mouse selections: com.-arg.+com. - 10 com.-arg.+arg. - 84 Cat. of comments: com.-arg.+com. - fav. com.-arg.+arg. - none	Comments: fav.	Comments: Abbrev.	Method of act. mech: sys. resp. ✓ Cat. of comments: fav. Usefulness of mech: Unsure	Digression: fav. Return: fav.

Table 7.4 Summary of subject 4's protocol

- **Subject 5**

Free-order syntax

No comments were made about being able to vary arguments or commands within Circuit II's interface.

The first time S5 encountered the "?" he correctly assumed that the system was waiting for him to specify a command action against the argument.

Deixis

S5 did not get from the system what he expected when requesting a component's specification as he thought he would be told whether the component was faulty or not. Although he did not object to the system's response when a component's specification was requested, he did not like computers to be conversational, rather he expected them to be factual.

Textual responses

S5 would have preferred numerical answers to be displayed as opposed to text and felt that the responses given were too verbose and contained a lot of redundant information.

Is that right?

S5 saw the activation of the "Is that right?" mechanism as a nuisance value and would have preferred to double click on a component as opposed to clicking on a "√" and the "sys. response". He thought that clicking on the "√" and then on the "sys.response" was better than clicking on the "sys.response" and then on the "√". Whether or not his actions could be considered as challenging the system's previous measurement would depend upon why he was asking the question. He said "I think it is an unnecessary question... what is the purpose of asking the question?". Despite these comments, S5 said "I find it good to get the information without having to reselect... without having to go back... deselecting..."

Conversational coherency

When changing context S5 thought that the window changing from green to red possibly signified a fault condition. Upon returning to the previous context S5 felt that the response gave him a sense of continuity in the interaction and thought that it would be more important in a larger circuit. He did not feel that it was harmful to return to the previous context with the original icons highlighted.

A summary of S5's responses within these categories is given in Table 7.5 below.

Free-order Syn.	Deixis	Text. resp.	Is That Right Mech.	Conv. Coh.
Mouse selections: com.-arg.+com. - 7 com.-arg.+arg. - 39 Cat. of comments: com.-arg.+com. - none com.-arg.+arg. - none	Comments: Unfav.	Comments: Short	Method of act. mech: √ sys. resp. Cat. of comments: fav. Usefulness of mech: fav.	Digression: No comment Return: fav.

Table 7.5 Summary of subject 5's protocol

- **Subject 6**

Free-order syntax

After correctly guessing how to take the voltage measurement of a component and then its current, S6 said that "it was easy to understand because the component visibly changed" to reflect this change in status which confirmed to him what he thought he was doing. S6 said that the system was intelligent enough to permit him to carry on taking voltage measurements of different components until he specified otherwise. He found being able to do this a useful feature because it meant that he did not have to keep on respecifying his intentions all the time.

The first time S6 encountered the "?" he correctly assumed that the system was waiting for him to specify a command action against the argument.

With inspection points he preferred the default mechanism supported by Circuit II because he knew where he stood with it. If an inferencing mechanism were used he might not agree with its decision to de-highlight a particular inspection point.

When combining commands and arguments in a hypothetical fault context to open-circuit a component, he viewed his actions as actually having faulted the component rather than doing so hypothetically. Not having noticed the window colour change from green to red, S6 was unaware that he was in a hypothetical fault mode, but when this was explained to him he agreed that his graphical statements were equivalent to asking a hypothetical 'If x then y' question.

Deixis

S6 did not think that the system gave him a meaningful response because it did not seem to shift with him. He did not like the message 'Anyway...' because it returned him to the previous measurement. As far as he was concerned, he was not thinking about the previous measurement he made only the one he wanted to take next.

Textual responses

S6 did not like having the system give him graded responses, rather he preferred a full textual response from the system. He did not think that it took much longer to read or scan the full textual response.

Is that right?

No comments were made concerning the preferred method of activating the 'Is that right?' mechanism. S6 did not feel that by using this mechanism he was challenging the system's previous measurement, rather he was trying to find out what it should be, compared to a normal working circuit. He thought that the mechanism was very useful and said "Well that was great. It was telling me exactly that what I had got there would be the same no

matter what happens to that particular component." He liked the way the information was presented to him, telling him exactly what he wanted to know although he would like to control when the information was removed from the screen.

Conversational coherency

S6 initially didn't notice that the window had changed colour when he changed context until it was brought to his attention. When it was, he correctly identified the change as a move to a faulted condition. He recognized that the normal working and fault modes were distinct contexts. Initially S6 thought that the textual response he was given upon return to a previous context was just the system trying to be clever. After a discussion he saw what the point to the previous context being displayed to him was. S6 felt that he would need to use the system for some time before he could give an opinion on it.

A summary of S6's responses within these categories is given in Table 7.6 below.

Free-order Syn.	Deixis	Text. resp.	Is That Right Mech.	Conv. Coh.
Mouse selections: com.-arg.+com. - 13 com.-arg.+arg. - 99 Cat. of comments: com.-arg.+com. - fav. com.-arg.+arg. - fav.	Comments: Unfav.	Comments: Full	Method of act. mech: √ sys. resp. Cat. of comments: fav. Usefulness of mech: fav.	Digression: Unsure Return: Unsure

Table 7.6 Summary of subject 6's protocol

- Subject 7

Free-order syntax

S7 thought it was fine that the system remembered how you got to a certain position in your measurement observations. He also liked the fact that it remembered the previous measurement he had taken because it gave him a nice option of introducing a fault into the system whilst looking at the

voltages. He thought this was a good feature. It was interesting, he said, that the system remembered his previous mouse selections.

Being able to latch onto a ground rail and then step through the other inspection points taking voltage readings with respect to it gave him the most useful measurements. S7 had a preference not just to root the probe to the ground connection but also to other inspection points and then step through them in a similar manner. He does not see that the current mechanism has any serious weaknesses and would prefer it to be kept simple.

The first time S7 encountered the "?" he correctly assumed that the system was waiting for him to specify a command action against the argument.

Deixis

S7 thought the responses given were meaningful although he did not like the word 'here' which is part of the phrase 'Anyway...' to draw his attention back to the previous measurement taken. He suggested rather than using the word it might be better to refer explicitly to the component or its parts. This aside, he said the response made sense.

Textual responses

S7 preferred the "Through the collector it's 6.29 milliamps" sort of response which falls between the full and the short expressions. He also felt that there was no need to display the full text associated with "If x then y" questions because the highlighted icons within the graphical display was explicit enough.

Is that right?

No comments were made concerning the preferred method of activating the 'Is that right?' mechanism. He did not see this mechanism as a means of challenging the system's previous measurement, rather he saw it as a

method of comparing it against what it would be in a normal working circuit. He said "What is useful is the '√'. I think the delay in the display of answers given by the mechanism is too long especially if you start getting impatient". This aside he "found using it very convenient."

Conversational coherency

From the textual response S7 was able to deduce that the system was telling him what he previously did in this context prior to the digression. He called the objects within the green window (the normal working context) the 'real world' and those within the red window 'imaginary.' Returning to a previous context with the previously highlighted icons was helpful because he did not have to start from a completely neutral position. S7 also felt that moving between contexts in the way supported by Circuit II gave him a sense of continuity. The window colour was important too for it gave him a sense of continuity because it allowed him to know which context he was in.

A summary of S7's responses within these categories is given in Table 7.7 below.

Free-order Syn.	Deixis	Text. resp.	Is That Right Mech.	Conv. Coh.
Mouse selections: com.-arg.+com. - 28 com.-arg.+arg. - 53 Cat. of comments: com.-arg.+com. - fav. com.-arg.+arg. - fav.	Comments: fav.	Comments: Abbrev.	Method of act. mech: √ sys. resp. Cat. of comments: fav. Usefulness of mech: fav.	Digression: fav. Return: fav.

Table 7.7 Summary of subject 7's protocol

- **Subject 8**

Free-order syntax

When taking measurements S8 made no comments about varying the argument portion of his mouse selections. However he liked the mechanism for requesting voltage and current readings which he thought "was fair enough".

The first time S8 encountered the "?" he correctly assumed that the system was waiting for him to specify a command action against the argument.

Deixis

The responses given by Circuit II when requesting a component's specification all seemed "perfectly clear and reasonable". S8 felt that the system did not need to revert back to remembering the voltage across the previous component once the specification of the component had been given.

Textual responses

S8 thought that the textual responses displayed were logical given the graphically highlighted objects, but said that the voltage across a resistor would normally be represented by, for example, "Vr 1k2 = 7.5290". Whilst S8 thought that the textual response was clear he was unsure if it still would be if the text was abbreviated. He agreed that the textual responses given were an accurate reflection of what was being displayed graphically.

Is that right?

Having already asked for current, S8 thought that he should just be able to click on to the "√" to compare the measurement given against its normal working value. He did not comment on whether he viewed his actions as challenging the system's previous response.

Conversational coherency

When moving from a normal to a faulted context S8 correctly deduced that the red window indicated a fault condition. He found that the textual response upon a return to the previous context gave him a sense of continuity. Rather than taking him back to scratch he felt that it had taken his thought processes back to where he was prior to the digression.

A summary of S8's responses within these categories is given in Table 7.8 below.

Free-order Syn.	Deixis	Text. resp.	Is That Right Mech.	Conv. Coh.
Mouse selections: com.-arg.+com. - 16 com.-arg.+arg. - 28 Cat. of comments: com.-arg.+com. - none com.-arg.+arg. - fav.	Comments: fav.	Comments: Short	Method of act. mech: *√. sys. resp.* Cat. of comments: General Usefulness of mech: no comment	Digression: fav. Return: fav.

Table 7.8 Summary of subject 8's protocol

- **Subject 9**

Free-order syntax

S9 thought that it was reasonable that when altering a command (fault or measurement) within the interface that this change should be reflected in the currently highlighted component (argument). Taking a new measurement by selecting another component in favour of a previous one made sense when taking measurements of the circuit. It seemed a reasonable thing to do.

The first time S9 encountered the "?" he correctly assumed that the system was waiting for him to specify a command action against the argument.

Deixis

S9 did not use the deixis mechanism.

Textual responses

S9 had no objections to the interrupt-return mechanism but did not like the textual response. He thought that only the measurement should be given in conjunction with a component's highlighted inspection points.

Is that right?

S9 did not like the challenge-support box disappearing on its own, but rather would prefer to control its removal from the screen. No comments were made concerning the preferred method of activating the "Is that right?" mechanism. He did not think that his actions were challenging the

system's previous response. As he said "It is not a challenge, your asking something and what you expect is an answer." Despite this he did feel that it was a "very useful" mechanism.

Conversational coherency

When moving to a faulted context S9 was able to recognise that a fault condition had been introduced because the window (originally green) had changed to red. Although he found the re-instatement of the previous context and the accompanying textual response useful, he felt that there would be times when he would not like to see it come back to that. His reservations were more directed at the textual response than the highlighted icons. He felt that only the value should be displayed rather than the phrase "Well anyway...".

A summary of S9's responses within these categories is given in Table 7.9 below.

Free-order Syn.	Deixis	Text. resp.	Is That Right Mech.	Conv. Coh.
Mouse selections: com.-arg.+com. - 11 com.-arg.+arg. - 20 Cat. of comments: com.-arg.+com. - fav. com.-arg.+arg. - fav.	Comments: Not used	Comments: Short	Method of act. mech: V. sys. resp. Cat. of comments: fav. Usefulness of mech: fav.	Digression: fav. Return: fav.

Table 7.9 Summary of subject 9's protocol

7.5.2 Summary of subject protocols

To gauge the overall effectiveness of Circuit II's mechanisms comments made under the headings given in §7.5.1 were categorized as described below.

Free-order syntax

In section 6.2 the free-order syntax supported by Circuit II was discussed. This seeks to give users a greater degree of flexibility in posing questions

rather than being constrained by one particular type of rigid syntax. Categories 1 and 2 are associated with the command-argument+argument and command-argument+command syntax respectively. In particular, these categories contain data about (a) verbal comments made by a subject when using the syntax which is sub-categorized as being either (i) favourable (ii) unfavourable (iii) Unsure or (iv) no comment when none has been made.

A subject's competence with Circuit II's graphical syntax is measured by the number of correct mouse selection sequences performed and interpreted correctly. Of the 160 mouse selections made within category 1 and 441 within the category 2 (see table 7.10), all users were competent at using this form of syntax with no problems being reported.

Mouse Clicks made in each category	
Cat 1: Com.-Arg.+Com. act.	Cat2: Com.-Arg.+Arg. act.
160	441

Table 7.10 Activation of categories 1 and 2

These findings are further supported by users' verbal comments which show that in both categories seven of the nine subjects made favourable comments about the free-order syntax (see tables 7.11 and 7.12). The two remaining subjects did not comment on their use of the syntax.

Categorization of comments on Com.-Arg.+Com. act.			
Favourable	Unfavourable	Unsure	No Comment
7/9 (77.7%)	0	0	2/9 (22.3%)

Table 7.11 Category 1: analysis of comments

Also discussed in section 6.2 was the question-mark displayed by Circuit II as a means of indicating to a user that some further selection (a command, for example, 'measure voltage') is required when a component (an

argument) is activated first. Analysis of the data has shown that seven users correctly deduced, without assistance, what the question-mark implied, whereas two users had to be told its meaning.

Categorization of comments on Com.-Arg.+Arg. act.			
Favourable	**Unfavourable**	**Unsure**	**No Comment**
7/9 (77.7%)	0	0	2/9 (22.3%)

Table 7.12 Category 2: analysis of comments

Providing users with a flexible means of posing a question by either clicking on an argument followed by a command in any order saves users from having to remember which way round an icon should be activated and getting it right only fifty percent of the time. The results of the study have shown that subjects could use and understand Circuit II's free-order syntax which removed from them the need to remember in which sequence icons should be activated.

As well as having this flexibility users responded positively to the fact that the system could track their change of mouse selections, corresponding to a command or an argument variation, because it gave them the sense that the system knew what they were doing. Because the number of subjects who took part in the trials was small the evidence is inconclusive, but it does seem to suggest that mouse selections which perform analogous roles to anaphora, ellipsis and deixis used in natural language, can provide users with a more economical way of making requests than is the case in those interfaces which employ a more rigid syntax.

Deixis

The use of Deixis in everyday language, as discussed in Section 6.3.3, is an important means by which we refer to an object to disambiguate it from others, or to refer to the same object but mean different things. Given the

importance of this mechanism in natural language, its graphical counterpart, modelled in Circuit II, is an attempt to establish the place of this more general form within a direct manipulation environment. Category 3 contains data associated with subjects' use of this mechanism.

The number of mouse selections associated within this category (13) was very low compared with those made in other categories. In all, six of the nine subjects used the mechanism thirteen times: four used it only once, one used it twice and the remaining subject used it seven times. None of the subjects who used this facility found any difficulty in activating it.

Subjects' verbal comments associated with their deictical mouse selections were mixed and reflected different opinions about the usefulness of the mechanism. Of the six subjects who used the mechanism, three gave favourable responses, two gave unfavourable responses and one was unsure about it (see table 7.13).

Categorization of comments on Deixis act.			
Favourable	**Unfavourable**	**Unsure**	**Not Used**
3/9 (33.3%)	2/9 (22.2%)	1/9 (11.2%)	3/9 (33.3%)

Table 7.13 Category 3: analysis of comments

Favourable comments made by users concerned the general idea of the mechanism itself. One of them did not like the wording of the response given by the system signalling a return to the previously interrupted context but felt that anything which drew his attention back to his original train of thought was good. Another user thought that whilst the return response to the previously interrupted context was meaningful it was not necessary to display this as the graphical representation was explicit enough. The third user did not see the necessity of returning to the last measurement taken after a component's specification had been given. This aside, he found using

the mechanism perfectly clear and reasonable.

Two users gave unfavourable comments about the deixis mechanism. One user did not think that the system 'kept up with him' as the return-response kept on referring him back to the previously taken measurement. As far as he was concerned he was thinking of the next measurement he was going to take, not the previous one. Rather than criticising the mechanism or response given, the second user, when in the random fault mode, expected the component specification to reveal the fault to him. In addition, he did not like computers to be conversational, rather he expected them to be factual. In his concluding remarks he felt that, from an electronic perspective, the deixis mechanism was not a valuable facility.

Only one user was unsure about the appropriateness of the system response, "Anyway, as I was saying...", when different measurements (voltage, current and resistance) were being taken. Under these circumstances, when a component's specification is requested, he felt that it might not feel as though a digression has occurred.

Circuit II has shown that a more general form of deixis can be modelled in a graphical environment and can be used, for example, to request a component's specification. It is not clear though, given that a previous measurement has taken place, whether asking for a component's specification should be considered to be a brief conversational digression from the main context of taking the measurement. The results of the study are inconclusive and further experimentation on this issue needs to be carried out.

Textual responses

Circuit II attempts to give a wide variety of responses which are appropriate to the mouse selections being made by users and have already been discussed in §6.4. In a natural language system like SOPHIE, textual

responses were the only means by which information was conveyed to users. This is in contrast to Circuit II, which uses a mix of text and graphics to impart information. Category 4 contains data associated with subjects' use of this mechanism which is summarized in table 7.14.

The results of the analysis show that there was no general concensus about how verbose the system should be when responding to users. Three users preferred an abbreviation of the full system response. One of the three felt that there was no need to display the full text associated with "If x then y" questions as the graphical display was explicit enough. The other two users just stated their preference for an abbreviated form of response but gave no reasons.

Four users preferred very short responses giving, for example, voltage, current and resistance measurements as opposed to having responses containing redundant information.

Only one user preferred the full response because he did not like the graded responses given to him by the system. As far as he was concerned, it did not take much longer to read the full line of text or to scan it for the measurement than an abbreviated one.

Whilst the remaining user generally knew what was going on in the system he felt that the responses given were messy although he did not specify in what way.

Categorization of System responses			
Full	**Abbrev.**	**Short**	**Unsure**
1/9 (11.1%)	3/9 (33.3%)	4/9 (44.5%)	1/9 (11.1%)

Table 7.14 Category 4: analysis of comments

The general indication to be drawn from this analysis is that users do not

wish to be presented with textual information which is already explicit in a graphical form. The results of the data suggest that graded textual responses corresponding to graphical analogues of ellipsis and anaphoric-like mouse selections do not enhance the naturalness of the interaction between a user and the system. Rather, abbreviated responses in answer to questions which are clearly displayed graphically should be given thus allowing users to make their own linguistic association, if any, between the two. Equipment used by electronic engineers normally gives factual information concerning the state of a component or device and may account for subjects' preference for some abbreviated form of response.

Is that right?

Within a hypothetical or random fault context the measurements taken need to be compared against their normal working values. SOPHIE users compared such values by typing 'Is that right?' whereas Circuit II provides its users with a graphical means of doing so as described in §6.4.3. Category 5 contains data associated with subjects' use of this mechanism which is summarized in tables 7.15, 7.16 and 7.17.

All nine subjects used the mechanism at least once although only three specifically commented on the activation mechanism. Analysis of mouse selections show that for the two forms of syntax supported, which can activate the mechanism, users preferred the "√-sys.response" syntax which was used 52 of the 62 times. The "sys.response-√" syntax was only used 10 times. Regardless of the preferred syntax none of the users reported any difficulty in activating the mechanism. These results are shown in table 7.15.

Method of activating "Is that right?" mechanism	
∜ - sys. response	sys. response - ∜
52/62 (83.9%)	10/62 (16.1%)

Table 7.15 Activating "Is that right?" mechanism

Only one of the nine users thought that by activating the mechanism he was challenging the system's previous response. Three users viewed their actions as a means of clarifying what the correct measurement should be rather than a challenge. The five remaining users did not indicate what they thought about the mechanism (See table 7.16).

Categorization of comments on 'Is that Right?' mechanism		
Challenge	Comparison	No Comment
1/9 (11.2%)	3/9 (33.3%)	5/9 (55.5%)

Table 7.16 Category 5: analysis of comments

With respect to the usefulness of providing such a facility within a graphical environment, five out of nine users thought it was a very useful and powerful facility. One user was unsure about how useful it would be in reality and the remaining three subjects did not pass any comments (See table 7.17).

Usefulness of 'Is that Right?' mechanism		
Useful	Unsure	No Comment
5/9 (55.6%)	1/9 (11.1%)	3/9 (33.3%)

Table 7.17 Usefulness of "Is that right?" mechanism

General comments made by three users concerned the removal of the challenge-support window from the screen. Rather then operating on an automatic delay mechanism, as it does at the moment, they thought that the window should remain displayed until they chose to remove it by clicking on it with the mouse.

The hypothesis that the phrase "Is that right?" could be viewed as a challenge to some previous measurement given by the system is not supported by the data. The results suggest that such a user-posed question is an attempt to establish what the measurement given by the system would be in a normal working context. The major advantage of the mechanism was that a measurement taken in a hypothetical or random fault context could be compared against the same in a normal working context without having to return to it. It was generally thought that the way of activating the mechanism involved too many mouse selections and could be simplified.

Conversational coherency

Cue phrases, as described in §6.5, are a means used by conversational participants to signal a suspension of a topic being discussed in favour of a related but tangential one before eventually resuming the interrupted topic again. Modelling graphical equivalents of cue phrases in Circuit II is an attempt to extend and support the notion of conversational coherency between related contexts in a graphical environment. Category 6 contains data associated with subjects' comments made when activating the interruption-return mechanism.

When pressing the second mouse button, either over a component (an argument) or a specified use (a command) which corresponds to the clue word "Incidentally..." when asking a hypothetical "If x then y" question, four out of six users said that the text responses were very clear and reflected what was graphically displayed on the screen. The remaining two users liked the way the "If x then y" questions were graphically displayed. One user did not feel that the textual response which accompanied the "If x then y" question had any meaning because, at this time, he had nothing against which he could compare it. He was also not sure why he would ask such a question. Table 7.18 summarizes these results.

Activating the clue word 'Incidentally'		
Clear Textual Response	**Good Graphical Display**	**No Comment**
4/9 (44.5%)	2/9 (22.2%)	3/9 (33.3%)

Table 7.18 Digression to a related context

Activating a clue word suspends the current context in favour of the newly activated hypothetical fault context which is signalled to the user by the window changing from green to red as discussed in §6.5.3. Analysis of the data has shown that seven of the nine users correctly deduced that they had changed context and identified the changing of the window colour, from green to red, as the reason. Of the other two users, one, upon returning to the previously interrupted context and reading the system response, said that he had come back to where he had started. The remaining user thought the window changing colour was a thoughtful way of alerting him to something.

Six out of seven users thought that the return mechanism gave them a sense of continuity in that it brought them back to where they left off in the previously interrupted context. Having the icons in this context re-instated assisted in focussing their minds back onto their original train of thought. Users liked Circuit II's ability to support resumptions in this way because it offers an advantage over conventional menuing systems by maintaining the state of the previously interrupted context until a resumption is signalled. One user felt that there would be times when they would not wish to see the system display the response 'Anyway...' and would prefer only the measurement to be given. This aside he did think that the mechanism was useful. Both subjects who were unsure about the mechanism felt that they were not familiar enough with it and would need to use it for some time before being able to pass an opinion (See table 7.19). Supporting cue phrases which enable users to digress to a hypothetical fault context whilst temporarily suspending the previous one was well received by users. By

supporting such digressions Circuit II has demonstrated that users' intentions, as expressed through their mouse selections, can be tracked and responded to meaningfully over an extended interaction.

Activating the clue word 'Anyway'		
Gave a sense of continuity	Useful mechanism	Unsure
6/9 (66.7%)	1/9 (11.1%)	2/9 (22.2%)

Table 7.19 Resumption of a previously suspended context

By making the change of context explicit to users, the results of the study have also shown how the same mouse selections made in a hypothetical fault context can have an entirely different meaning in the random fault context. By contextualization therefore, some of the different meanings associated with combinations of selected icons can be made explicit to the user. An additional advantage of such an approach is that, providing that the icons are well designed it becomes possible to model a wide range of graphical discourse with a limited set of icons.

7.6 Summary

The overall results of the observation trials are encouraging because they have shown that the features supported by Circuit II can be used and understood by users.

Supporting a free-order syntax has shown that it saves users time and convenience in terms of economy of mouse selections made. This flexibility of expression allows users to vary commands and arguments giving then the sense that their mouse selections are being tracked both within and between contexts. Free-order syntax it is argued, is a novel feature which makes an important contribution to the usability of direct manipulation interfaces. As such it is an improvement over direct manipulation interfaces which support more rigid forms of syntax. It is argued that the

empirical study has provided evidence to support the claim that, Circuit II, in extended interaction sequences, matches the brevity, naturalness and convenience that SOPHIE achieved through its use of surface linguistic features.

Being able to point at a component and mean different things frees users from the use of pull-down menus and supports a more general form of deixis, like that found in natural language. Although use of the mechanism was limited, its support is an improvement on those direct manipulation interfaces within which pointing only means one thing.

The degree of verbosity which should be displayed by Circuit II is unclear, as there was no concensus among the subjects. The solution to this problem would seem to develop some mechanism by which subjects could adjust the verbosity of the system according to their own preference.

The evidence pertaining to the "Is that right?" mechanism seems to suggest that it should not be viewed as challenging the system response to a question. Rather, it should be viewed as a means by which an answer given to a question in a faulted context can be compared to the same question posed in a normal context. Although this mechanism is very useful, activating it involves too many mouse selections and an alternative method of activation needs to be explored.

On the whole users responded positively to the fact that the system could track their change of mouse selections between contexts using graphical equivalents of cue phrases. The majority of subjects were aware when they had moved to a faulted context and when they had returned. Despite problems with the wording of the "return" response, re-instating the display as it was prior to the digression enabled subjects to resume their actions in this context in a meaningful way.

The next and final chapter examines what has been achieved in this research work and how it might be explored further.

Conclusions

8.1 Introduction

The aim of the research work reported in this thesis was to investigate the possibility of modelling natural language and conversational coherency within a direct manipulation interface. Two systems, Circuit I and Circuit II were developed to show how this could be achieved. These systems model natural language questions, typical of those posed in the electronics domain, graphically. Through the interface, a respresentation of an electronic circuit, deixis, ellipsis, anaphoric-like actions and topic shifts are supported. The need to support conversational features within a direct manipulation environment was identified from the literature on interactional styles employed by current interfaces. The remaining part of this chapter summarizes the achievements of the research work, outlines the contribution of the research in various areas, discusses the limitations of the research work and indicates some directions for further work.

8.2 Achievements

The achievements of this research are summarized below:

- A review of the literature on the main interactional styles commonly employed in computer interfaces which identifies the limitations of each approach.
- Observational evaluation of Circuit I using the results as a basis for the design of a second model.
- Re-implementation of SOPHIE I.
- Observational evaluation of Circuit II.

A review of the literature on the main interactional styles commonly employed in computer interfaces which identifies the limitations of each approach.

The review of literature conducted has sought to give an overview of the main interactional styles which are widely used in current computer interfaces. An in-depth summary of SOPHIE's natural language processing capabilities is given as is the work of Reichman's on conversational coherency. These summaries demonstrate the power of natural language as a communication medium which has applicability in a direct manipulation environment. The merging of these two summaries suggested a specification for modelling Human Computer Interaction on conversation. In particular, surface linguistic features (anaphora and ellipsis) which cannot be presented via direct manipulation should be meshed smoothly with direct manipulation style design as far as possible.

Observational evaluation of Circuit I using the results as a basis for the design of a second model.

Evaluating Circuit I has shown that when surface linguistic features are modelled within a graphical environment they facilitate the interaction process in a way analogous to their functions in natural language. This is demonstrated by users' mouse selections which are more economical when performing a particular action than would otherwise be the case in an interface without these features. The observational study highlighted the fact that the model did not support separate contexts for normal and faulted circuit behaviour. This knowledge was used in conjunction with the literature summaries as input to the design stage of the second model, Circuit II.

Re-implementation of SOPHIE I.

Despite attempts to resurrect the original SOPHIE I, this proved impossible. To substantiate the hypotheses presented in this thesis it was necessary to re-implement the major parts of SOPHIE I so that the graphical representation of Circuit II's audio video amplifier circuit could be substituted in place of SOPHIE's natural language interface. Only by doing this could Circuit II graphically model a representative range of SOPHIE dialogue which could be compared and constrasted in terms of economy of expression and performance. The re-implementation, involving the construction of SOPHIE's specialists, interfacing these to MITEYSPICE, and the design, construction and interfacing of Circuit II's graphical interface took place over a nine month period.

Observational evaluation of Circuit II

The evaluation of Circuit II has provided evidence to support the claim that, in extended interaction sequences, the brevity, naturalness, and convenience that SOPHIE achieved through its use of anaphora and ellipsis can be achieved in a direct manipulation environment. The evaluation has also shown how combinations of icons representing more than one meaning can be resolved through contextualisation in a window environment. In combination with the free-order syntax and a more general form of deixis supported by Circuit II, this extends the range of SOPHIE dialogue which can be modelled using a limited set of icons representing domain objects and concepts.

8.3 Contributions

This section examines the contributions of these achievements to the area of interface design.

8.3.1 A contribution to interface design

There are four areas in which a specific contribution has been made:

- Development of Circuit I which demonstrates how graphical analogues to ellipsis and anaphoric like actions may be supported in a direct manipulation interface using free-order syntax.
- Development of Circuit II which extends the range of free-order syntax supported in Circuit I to support two different types of command-arguments.
- Supporting a more general form of deixis in Circuit II.
- Supporting topic shifts in Circuit II's direct manipulation interface.

Development of Circuit I which demonstrates how analogues to graphical ellipsis and anaphoric like actions may be supported in a direct manipulation interface using free-order syntax.

The literature on SOPHIE indicated that the success of its natural language processing capabilities largely came from its ability to support surface linguistic features (anaphora and ellipsis) expressed by users during an interaction. The design of Circuit I was based on the way that electronic engineers troubleshoot an electronic circuit using, for example, a multimeter. The success of this model lies in its ability to demonstrate how supporting a free-order syntax allows users to express commands analogous to ellipsis, and anaphoric-like expressions within a direct manipulation interface.

Development of Circuit II which extends the range of free-order syntax supported in Circuit I to support two different types of command-arguments.

The free-order syntax of Circuit I supports a single command-argument structure which can be used to take various measurements within the circuit. Circuit II extends this free-order syntax with an additional command-argument structure to enable different faults to be applied to components. Like Circuit I, Circuit II's free-order syntax supports the

expression of commands which perform analogous roles to anaphora and ellipsis. This additional command-argument structure greatly increases the range of SOPHIE dialogue which can be modelled and the usability of Circuit II.

Supporting a more general form of deixis in Circuit II.

As has been discussed in §1.0 and §6.0, the use of direct manipulation techniques employ deixis. However, all pointing in these types of interface is ambiguous, and has to be combined with some other type of information to decide what is meant. Many direct manipulation interfaces have a rigid convention so that pointing only means one thing. The form of deixis supported by Circuit II is modifiable, so that pointing can mean one of three things; a measurement request, a component modification request, or a specification request. This, it is suggested, more closely approximates general deixis found in natural language.

Supporting topic shifts in Circuit II's direct manipulation interface.

A rather neglected aspect, but vital, is how to imitate, in computers, the apparently effortless shifts that people make between topics. Circuit II, like SOPHIE supports three different circuit contexts: normal circuit behaviour, a circuit with an unknown fault, and circuits with user-hypothesised faults. Whilst SOPHIE used explicit commands to move between these contexts, Circuit II exploits Reichman's notion of natural language cue phrases, for example, "by the way", re-implemented in the direct manipulation style to make such transitions smoother and more automatic.

8.4 General conclusions

Some general conclusions which can be drawn from this research are given below.

By taking features of natural language, namely, anaphora, ellipsis, deixis, conversational moves for interruption and resumption of a context and using them to invent analogues in direct manipulation interfaces greatly improves their usability.

For the current types of Intelligent Tutoring Systems there is no loss of functionality if you move from a natural language interface to an appropriately designed graphical interface.

The success of SOPHIE's interface depended upon two things. First it allowed users to express themselves in the language of the domain. SOPHIE used natural language as an input medium to achieve this whilst Circuit II uses diagrams. Second, SOPHIE permitted abbreviated input whose meaning was obvious in context, but whose form was structurally incomplete. It achieved this using anaphora and ellipsis whilst Circuit II uses a free-order syntax to allow general ellipsis.

8.5 Limitations

This section examines the limitations of the research presented in this thesis.

Observational studies.

The two observational studies conducted with the models developed, Circuit I and Circuit II, used a small number of subjects (Circuit I: n=4; and Circuit II: n=9). The results obtained were positive and strong indicators that the support of conversational features enhanced the usability of the interface. However, a more detailed empirical study needs to be conducted to further support the findings presented in §7.5, using a minimum of fifty subjects.

Circuit I.

The prototype model of Circuit I is very simple using only arrays of data in conjunction with electronic formulae to derive answers. It can only perform simple calculations.

Circuit I is limited in that only pre-determined faults can be applied to the components of the circuit using pull-down menus. Also, it does not support transitions between a faulted and non-faulted context and thus does not differentiate between the two states. Only a small portion of SOPHIE dialogue could be modelled because of the limited types of components modelled.

Circuit II

Circuit II, unlike SOPHIE, has no fault propagation specialist which can report on the sequence of component failures as a fault propagates through the circuit.

At present Circuit II does not have an explanation specialist which is capable of suggesting a hypothesis to match a set of measurements taken by a user.

The circuit simulator MITEYSPICE used by Circuit II needs to be modified to reside permanently in main memory rather than being loaded each time it is invoked.

Due to time restrictions Circuit II does not fully implement all the features of SOPHIE I.

Only one circuit of moderate complexity is modelled by Circuit II.

The design of multimeter icons in Circuit II is crude and requires re-designing.

Only one fault may be inserted into Circuit II at a time.

Only one question can be presented to Circuit II at a time.

8.6 Further work

The three preceding sections in this chapter have discussed the achievements, contributions made and the limitations of the work presented in this thesis. This section discusses how this work might be extended. Three areas which could be developed further are:

- extension of observational study
- extension of graphical interface
- supporting more conversational features

Extension of observational study.

1) The number of subjects used in the observational study was sufficient for the design of the two models because only expert electronic troubleshooters were used. Each subject gave a wealth of quality information which more than compensated for the lack of subjects. However, in order to further prove the notion of conversational coherency in a graphical interface a more detailed empirical study with a larger number of subjects and experience will have to be undertaken to show that the results obtained are statistically significant.

Extension of graphical interface.

1) The graphical interface at present displays a range of icons representing the electronic circuit, the multimeter, faults to be applied to components and repair equipment. With the exception of the circuit icons, the others are crude and an empirical study needs to be conducted to establish how they could be made more meaningful. One way of doing this would be to interview a large number of electronic engineers to try and establish if, for example, there was a common iconic symbol for open-circuit types of faults.

Such icons could be used to replace the existing ones used by Circuit II making their function more apparent.

2) All icons highlighted by mouse actions are accompanied by textual responses which are graded according to the type of action being performed, for example, anaphora, ellipsis and deixis. The degree of verbosity which should be displayed by the system is unclear as an analysis of the protocols collected were inconclusive. Some subjects preferred verbose responses whilst other preferred abbreviated or numerical ones. An empirical study could be conducted to see if a general consensus amongst electronic engineers emerged about the degree to which they would like a computer system to be verbose and why. Depending upon the results, Circuit II's measurement library could then be modified to display only, for example, verbose responses.

Supporting more conversational features.

Reichman's theory of discourse processing, although criticised in a natural language setting, has merit in a graphical context and deserves consideration. Circuit II supports multiple window contexts, which, because of limited screen space, are not displayed simultaneously to users. Circuit II's interface has made some progress in implementing Reichman's (1986) work by:

a) providing underlying support mechanisms which define individual contexts and the relations between these contexts and the objects they contain;

b) making explicit the dependencies and interrelationships between these contexts using graphical techniques.

One way in which this work could be extended is by conducting an empirical study which collects and analyses electronic troubleshooting protocols with a view to establishing the range of conversational moves made by

troubleshooters. The results of this analysis could be used to develop and extend a conversational module capable of supporting this range of possible moves graphically. To support this, some kind of language would need to be developed to note the types of activity shifts being made (Reichman, 1986). A further empirical study could then be carried out to establish if the type of shift indicated by the module corresponded to that the one that the user thought they were performing.

8.7 Summary

SOPHIE was a landmark Intelligent Tutoring System. It contained many important ideas, but as with many innovative systems, it is not possible to be sure which of its many features were important to its overall success. One of these was its natural language user interface. Today, most successful user interfaces use direct manipulation and graphics instead of natural language, but many researchers still argue that truly natural and inviting user interfaces should be based upon the latter. Thus in both Intelligent Tutoring Systems and Human Computer Interaction fields, there is an argument that is fairly often heard in favour of natural language user interfaces.

This thesis has explored this question by re-implementing SOPHIE with a graphical direct manipulation interface instead of a natural language one, while retaining or improving its standard of usability. It began by analysing the features that seemed to have been central to SOPHIE's usability. These, it has been argued, were not so much an ability to accept well formed complete English sentences, as an ability to accept and interpret correctly a wide variety of abbreviated inputs: technically, to deal with anaphora and ellipsis. In fact its choice of semantic grammars as a technical approach was made in order to support these "pragmatic" features. These features can be argued to be part of conversational rather than narrowly linguistic

competence: that is, they support extended interaction, rather than the expression of isolated propositions.

Two direct manipulation interfaces were implemented and tested, one a pilot and one a fairly full re-implementation of SOPHIE. They employ a technique of free-order syntax that allows users to specify the components of a full command in any order. Technically, any use of direct manipulation employs deixis, and the free-order syntax allows completely general ellipsis (any component not newly specified are inherited as defaults from the last command executed). This achieves in extended interaction sequences, the brevity, naturalness, and convenience that SOPHIE achieved through allowing ellipsis and anaphora in natural language. This claim is supported in detail by a) the analysis and comparison of extensive examples from SOPHIE and these new programs, b) the generation of English output as part of the programs' responses with the linguistic features at issue (ellipsis, anaphora) and c) detailed testing of the programs on domain experts (electronics troubleshooting technicians), looking for evidence of naturalness of both the direct manipulation input language and both direct manipulation and English outputs, and also the overall acceptability of the interactions.

The free-order syntax technique is at best little used currently in user interfaces, and yet, as has been shown here, it saves users time and convenience. Thus considering key linguistic features of a natural language user interface has led to a novel approach argued for in this thesis: that language or at least conversation offers an important source of phenomena which can be drawn upon to improve user interfaces; but which although taken from natural language do not need to be implemented in natural language interfaces; instead analogues for them may be constructed within direct manipulation interface styles.

This more general approach was further developed in the field of topic shifts. SOPHIE had in effect three topics (normal circuit behaviour, a circuit with an unknown fault, and circuits with user-hypothesised faults), but managed switches between them clumsily via explicit commands. In direct manipulation interfaces the analogues of topics are windows, but in most interfaces this correspondence is not well matched, leading to defects in usability. Drawing on Reichman's work, shifts between topics which are managed in natural language by cue phrases such as "by the way" were handled in the direct manipulation style re-implementaion in a smoother way, and the naturalness of this "translation" to the direct manipulation idiom was tested on users.

References

Akscyn, R.M.; and McCracken, D.L. (1988) KMS: A distributed hypermedia system for managing knowledge in organizations. *Communications of the ACM*, 31 (7), pp. 820-835.

Algers, J.; Benyon, D.; Davies, G.; Dobson, S.; Head, F.; Keller, L.; Passey, L.; Preece, J.; and Rogers, Y. (1990) *A Guide to Usability, Usability Now!* The Open University, Milton Keynes, U.K. BPCC Wheaton Ltd, Exeter.

Allen, J. (1987) *Natural Language Understanding.* The Benjamin/Cummings Publishing Company, Inc.

Baecker, R.M. (1980a) Towards a Characterization of Graphical Interaction. In Guedj, ten Hagen, Hoopgood, Tucker & Duce (Eds.), *Methodology of Interaction,* Amsterdam: North Holland, pp. 127-147.

Baecker, R.M. (1980b) Human-Computer Interactive Systems: A State-of-the-Art Review. In Koler, Wrolstad and Bouma (Eds.), *Processing of Visible Language* 2, New York: Plenum Press, pp. 423-443.

Blandford, A. (1991) *Design, Decisions and Dialogue.* Ph.D. Thesis, The Open University, Milton Keynes, U.K.

Bobrow, G.B.; Kaplan, R.M.; Kay, M.; Norman, D.; Thompson, H.; and Winograd, T. (1986) GUS, A Frame-Driven Dialog System. Grosz, B.J.; Jones, K.S.; and Webber, B.L. (Eds.) (1986) In *Readings in Natural Language Processing*, Morgan Kaufmann Publishers, pp. 595-603.

Bolt, R.A. (1984) *The Human Interface: Where People and Computers Meet,* Boston: Lifetime Learning Publications.

Bolt, R.A. (1981) Gaze-Orchestrated Dynamic Windows. *Computer Graphics* , 15(3), pp. 109-119.

Brown, J.S.; Burton, R.R.; and de Kleer, J. (1982) 'Pedagogical, natural language, and knowledge engineering techniques in SOPHIE I, II, and III' in Sleeman, D.H.; and Brown, J.S. (Eds.) *Intelligent Tutoring Systems,* Academic Press, London, pp. 227-282.

Brown, J.S.; Rubinstein, R.; and Burton, R.R. (1976) Reactive Learning environment for computer-assisted electronics instruction. *BBN Report 3314.* Bolt Beranek & Newman Inc., Cambridge, Massachusetts.

Brown, J.S.; Burton, R.R. (1975) 'Multiple representation of knowledge for tutorial reasoning' in Bobrow, D.; and Collins, A. (Eds.) *Representation and Understanding: Studies in Cognitive Science,* Academic Press, New York, pp. 311-349.

Brown, J.S.; Burton, R.R; and Bell, A.G. (1974) SOPHIE: a sophisticated instructional environment for teaching electronic troubleshooting. *BBN Report no. 2790.* Bolt Beranek & Newman Inc., Cambridge, Massachusetts.

Brown, J.S.; Burton, R.R; and Zdydel, F. (1973) A model-driven question-answering system for mixed-initiative computer-assisted instruction. *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 3, pp. 248-257.

Burton, R.R.; and Brown, J.S. (1979) Towards a natural language capability for computer-assisted instruction. Reprinted in Grosz, B.J.; Jones, K.S.; and Webber, B.L. (Eds.) (1986) *Readings in Natural Language Processing,* Morgan Kaufmann Publishers, Inc, pp. 605-625.

Burton, R.R. (1976) Semantic grammars: an engineering technique for constructing natural language understanding systems. *BBN Report 3453 (ICAI Report 3).* Bolt Beranek and Newman Inc., Cambridge Massachusetts.

Burton, R.R. (1975) *Semantically-centered parsing.* Doctoral dissertation, University of California, Irvine, California.

Buxton, W.; Fiume, E.; Hill, R.; and Woo, C. (1983) Continuous Hand-Gesture Driven Input. *Proceedings of Graphic Interface '83,* pp. 191-195.

Carbonell, J.R. (1970) AI in CAI: an artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems,* Vol. 11, no.4, pp. 190-202.

Card, S.K.; and Henderson, D.A. Jr. (1987) A multiple virtual-workspace interface to support user task switching. In Proceedings of CHI +GI,

Human Factors in Computing Systems and Graphics Interface, ACM, pp. 53-59.

Card, S.K.; Pavel, M.; and Farrel, J.E. (1985) Window-based Computer Dialogues. *Human-Computer Interaction - Interact '84,* Amsterdam: North Holland, pp. 239-243.

Cohen, P.R.; Dalrymple, M.; Moran, D.B.; Pereira, F.C.N.; and Sullivan, J.W.; Gargan, R.A.; Schlossberg, J.L.; Tyler, S.W. (1989) Synergistic Use of Direct Manipulation and Natural Language. *CHI '89 Conference Proceedings.* pp. 227-233.

Collins, A.; Warnock, E.H.; Aiello, N,; and Miller, M.L. (1975) Reasoning from incomplete knowledge. In Bobrow, D.; and Collins, A. (Eds.) *Representation and Understanding: Studies in Cognitive Science,* Academic Press, New York, pp. 383-415.

Crystal, D. (1980) *A First Dictionary Of Linguistics And Phonetics.* University Press, Cambridge.

Cypher, A. (1986) The Structure of Users' Activities. In Norman, D.A.; and Draper, S.W. (Eds.) *User Centered System Design,* Hillsdale, New Jersey.: Lawrence Erlbaum Associates, pp. 243-263.

Delin, J. (1986) Getting Computers to Talk like You and Me. Book Reviews. *Computational Linguistics,* Vol 12 (4), October-December.

Draper, S.W. (1986) Applying Features of Conversation to HCI. Paper Presented at the *British Psychological Society* conference, April, 1986.

Ericsson, J.A.; and Simon, H.A. (1980) Verbal Reports as Data. *Psychological Review* 87(3), pp. 215-251.

Esterby, R.S. (1970) The Perception of Symbols for Machine Displays. *Ergonomics* 13(1), pp. 149-158.

Gittens, D. (1986) Icon-Based Human-Computer Interaction. *International Journal of Man-Machine Studies,* 24, pp. 519-543.

Grosz, B.J.; Jones, K.S.; and Webber, B.L. (Eds.) (1986) *Readings in Natural Language Processing,* Morgan Kaufmann Publishers.

Grosz, B.J. (1981) Focusing and Description in Natural Language Dialogues. *In Elements of Discourse Understanding,* Joshi, A.K.; Weber, B.L.; and Sag, I.A. (Eds.), Cambridge University Press, pp. 84-105.

Grosz, B.J. (1977) *The representation and use of focus in dialogue understanding.* Ph.D. Thesis, University of California, Berkeley.

Harris, L. (1977) User Oriented Database Query with the ROBOT Natural Language Query System. *International Journal of Man-Machine Studies* 9, pp. 697-713.

Hendrix, G.G.; Sacerdoti, E.; Sagalowicz D.; and Slocum, J. (1978) Developing a Natural Language interface to complex data. *ACM Transactions on Database Systems* 3, 2, pp. 105-147.

Hollan, J.D.; Hutchins, E.L.; and Weitzman, L. (1984) STEAMER: an interactive inspectable simulation-based training system. *AI Magazine,* Vol. 5, No. 2, pp. 15-27.

Hutchins, E.L. (1987) Metaphor For Interface Design. *NATO-sponsored workshop on Multimodal Dialogues Including Voice,* Venaco, Corsica, France. Available from Edwin Hutchins, Institute for Cognitive Science, C-015, University of California, San Deigo, La Jolla, CA 92093.

Hutchins, E.L.; Hollans, J.D.; and Norman, D.A. (1986) Direct Manipulation Interfaces. In Norman, D.A.; and Draper, S.W. (Eds.), Hillsdale, New Jersey: Lawrence Erlbaum Associates,*User Centered System Design,* pp. 87-124.

Jarvella, R.J. (1982) *Speech, Place and Action,* Wiley and Sons.

Jervell, H.R.; and Olsen, K.A. (1985) *Behaviour and Information Technology* 4(3), pp. 249-254.

Jones, S.; and Downton, A. (1991) Windowing Systems: high- and low-level design issues. In Downton, A (Ed.) *Engineering The Human-Computer Interface,* McGraw-Hill Book Company.

Jones, W.P.; and Landauer, T.K. (1985) Context and Self-Selection Effects in Name Learning. *Behaviour and Information Technology* 4(1), pp. 3-17.

Klein, E.; and Pineda, L.A. (1990) Semantic and Graphical Information. *Human-Computer Interaction - INTERACT '90,* pp. 485-491.

Klein, E. (1987) Dialogues with Language, Graphics and Logic. In CEC-DG XIII (Ed.) *ESPIRIT '87: Achievements and Impact,* North-Holland, Amsterdam, pp. 867-873.

Koler, P.A. (1969) Some Formal Characteristics of Pictograms. *American Scientist* 57(3), pp. 348-363.

Landauer, T.K.; and Nachbar, D.W. (1985). Selection from Alphabetic and Numeric Menu Trees Using a Touch Screen: Breadth, Depth, and Width. *Proceedings of CHI '85,* pp. 73-78.

Lee, J.; Kemp, B.; and Manz, T. (1989) Knowledge-Based Graphical Dialogue: A Strategy and Architecture. *In Proceedings of ESPIRIT '89.* Project no. 393. pp. 321-333.

Lodding, K.N. (1983) Iconic Interfacing. *IEEE Computer Graphics and Applications* 3(2), March/April 1983, pp. 11-20.

Loftin, R.B.; Wang, L.; Baffes, P.; and Hua, G. (1989) An Intelligent System for Training Space Shuttle Flight Controllers in Satellite Deployment Procedures. *Machine-Mediated Learning,* pp. 41-51.

Lyons, (1981) *Language, Meaning and Context.* Fontana Paperbacks.

Manes, S. (1985) Pushing Picture-Perfect Programs: Smash That Icon! *PC Magazine,* June, 1985,p. 64.

McCabe, D.J.; Childs, S.; and Clarke, C.J.T. (1988) *MITEYSPICE Circuit Simulator User Guide.* Those Engineers Ltd, Hampstead, London.

McCracken, D.L.; and Akscyn, R.M. (1984) Experience with the ZOG Human-Computer Interface System. *International Journal of Man-Machine Studies* 21, pp. 293-310.

Miller, J.R. (1988) The Role of Human-Computer Interaction in Intelligent Tutoring Systems. In Polson, M. C; and Richardson, J.J. (Eds.), *Foundations of Intelligent Tutoring Systems,* Hillsdale, New Jersey.: Lawrence Erlbaum Associates, pp. 143-189.

Miyata, Y.; and Norman, D.A. (1986) Psychological Issues in Support of Multiple Activities. In Norman, D.A.; and Draper, S.W. (Eds.) *User Centered System Design,* Hillsdale, New Jersey.: Lawrence Erlbaum Associates, pp. 265-284.

Nagel, L.W.; and Pederson, D.O. (1973) SPICE: Simulation Program with Integrated Circuit Emphasis. *Memorandum ERL-M382,* Electronics Research Laboratory, College of Engineering, University of California, Berkeley, April, 1973.

Nagel, L.W. (1971) Transient Analysis of Large Non-Linear Networks. *IEEE Journal of Solid State Circuits,* Vol SC-6, August, 1971.

Neal, J.G.; and Shapiro, S.C. (1988) Intelligent multi-media interface technology. In Sullivan J.W.; and Tyler S.W. (Eds.) *Architectures for Intelligent Interfaces: Elements and Prototypes,* Addison-Wesley, Palo Alto, C.A.

Negroponte, N. (1970) *The Architecture Machine: Towards a More Humane Environment,* Cambridge, Massachusetts: The MIT Press.

Newell, A.; and Simon, H.A. (1972) *Human Problem Solving.* Prentice Hall, Englewood Cliffs.

Nisbett, R.E.; and Wilson, T.D. (1977) Telling More Than We Can Know: Verbal reports on Mental Processes. *Psychological Review,* 84(3), pp. 231-259.

Norman, D. (1983). Design Principles for Human-Computer Interfaces. *Proceedings of CHI '83,* pp. 1-10.

Norman, D. (1981) The trouble with UNIX. *Datamation,* 27(11), November, 1981, pp. 139-332.

Perlman, G. (1985) Making the Right Choices with Menus. *Human-Computer Interaction - Interact '84,* Amsterdam: North Holland, pp. 317-321.

Perlman, G. (1984) Natural Artificial Language: Low Level Processes. *International Journal of Man-Machine Studies* 20, pp. 373-419.

Petrick, S.R. (1976) On Natural Language Based Computer Systems. *IBM Journal Research & Development*, 20, pp. 314-325.

Pineda, L.A. (1988a) A Compositional Semantics For Graphics. *Proceedings of The Sixth Annual EUROGRAPHICS Conference,* pp. 155-159.

Pineda, L.A.; Klein, E.; and Lee, J. (1988b) GRAFLOG: Understanding Drawings through Natural Language. *Computer Graphics Forum*,7, pp. 97-103.

Psotka, J.; Massey, L.D.; and Mutter, S.A. (1988) Intelligent Tutoring Architectures. Psotka, J.; Massey, L.D.; and Mutter, S.A. (Eds.) In *Intelligent Tutoring Systems Lessons Learned.* Lawrence Erlbaum Associates Publishers. pp. 403-408.

Reichman, R. (1986) Communication Paradigms for a Window System in Norman, D. A.; and Draper, S.W. (Eds.) *User Centered Systems Design*, Hillsdale, New Jersey.: Lawrence Erlbaum Associates, pp. 285-313.

Reichman, R. (1985) *Getting Computers to Talk Like You and Me*, The MIT Press, Cambridge, Massachussetts, London, England.

Reichman, R. (1984) 'Extended Person-Machine Interface' *Artificial Intelligence*, Vol. 22, pp. 157-218.

Reichman, R. (1981) *Plain Speaking: A Theory and Grammar of Spontaneous Discourse.* Ph.D Thesis. BBN Report no. 4681. Bolt Beranek & Newman Inc., Cambridge, Massachusetts.

Reichman, R. (1978a) Conversational Coherency. *Cognitive Science*, Vol. 2 pp. 283-327.

Reichman, R. (1978b) Conversational Coherency. *Technical report.* TR-17-78. Harvard University, Massachussets.

Rich, E. (1984) Natural Language Interfaces. *IEEE Computer* Sept. 1984, pp. 39-47.

Rich, E. (1983) *Artificial Intelligence,* McGraw Hill International Edition, pp. 295-341.

Robertson, G.; McCracken, D.; and Newell, A. (1981) The ZOG approach to man-machine communication. *International Journal of Man-Machine Studies* 14, pp. 461-488.

Robertson, J.; and Burns, A. (1985) A Dialogue Development System for the Design and Implementation of User Interfaces in Ada. *The Computer Journal,* Vol. 28, No. 1, pp. 22-28.

Rogers, Y. (1989) *Icons at the interface: their usefulness,* Butterworth & Co Ltd, pp. 105-117.

Scapin, D.L. (1982) Computer Commands Labelled by Users versus Imposed Commands and the Effect of Structuring Rules on Recall. *Proceedings of the Conference on Human Factors in Computing Systems,* pp. 17-19.

Schultz, J.R. (1986) A History of the PROMIS Technology: an Effective Human Interface. *Proceedings of the Conference on the History of Personal Workstations,* New York: ACM, pp. 159-182.

Schultz, J.R.; and Davis, L. (1979) The Technology of PROMIS. *Proceedings of the IEEE* 67(9), pp. 1237-1244.

Schute, J.; and Glaser, R. (1990) A Large-Scale Evaluation of an Intelligent Discovery World: Smithtown. In *Interactive Learning Environments,* 1(1), Ablex Publishing Corporation, pp. 51-77.

Shneiderman, B. (1986) *Designing the User Interface: Strategies for Effective Human-Computer Interaction,* Addison-Wesley Publishing Company.

Shneiderman, B. (1983) Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer,* Aug '83, pp. 57-62.

Shneiderman, B. (1982) The Future of Interactive Systems and the Emergence of Direct Manipulation. *Behaviour and Information Technology* 1(3), pp. 237-256.

Shneiderman, B.; and Mayer, R.E. (1979) Syntactic/Semantic Interactions in Programmer Behaviour: A Model and Experimental Results. *International Journal of Man-Machine Studies* 25(5), pp. 479-489.

Sidner, C.L. (1986) Focusing in the Comprehension of Definite Anaphora. Grosz, B.J., Jones, K.S., and Webber, B.L. (Eds.) In *Readings in Natural Language Processing*. Morgan Kaufmann Publishers. pp. 363-394.

Singer, R.A. (in press) Circuit II - A Conversational Graphical Interface. To appear in a special issue of the *International Journal Computers & Education* (Pergamon Press).

Singer, R.A. (1990) Graphical Treatment of Anaphora and Ellipsis in HCI. In journal of *Artificial Intelligence in Education* 2(1) pp. 79-97.

Singer, R.A. (1990) Graphical Treatment of Anaphora and Ellipsis in HCI: Empirical studies. Open University Publication: CITE Report No. 112.

Sisson.; Norwood.; Parkinson.; Stanley R.; and Snowberry, K. (1986). Considerations of Menu-Structure and Communication Rate for the Design of Computer Menu Displays. *International Journal of Man-Machine Studies* 25(5), pp. 479-489.

Sleeman, D.H.; and Hendley, R.J. (1979) ACE: a system which analyzes complex explanations. *International Journal of Man-Machine Studies* 11, pp. 125-144. (Reprinted in Sleeman, D.H.; and Brown, J.S. (Eds.) *Intelligent Tutoring Systems.* Academic Press, London.)

Smith, D.C.; Irby, C.; Kimball, R.; Verplank, B.; and Harslem, E. (1982) Design of the Star user interface. *Byte*, 7(4), pp. 242-282.

Tennant, H.R.; Ross, K.M.; Saenz, R.M.; Craig, W.T.; and Miller, J.R. (1983). Menu-based natural language understanding. In *Proceedings of the 21st Annual Meeting of the ACM.* New York: Association for Computing Machinery. pp. 151-158.

Tesler, L. (1981) The Smalltalk Environment. *Byte* 6(8). August 1981, pp. 90-147.

Thacker, C.P.; McCreight, E.M.; Lampson, B.W.; Spraill, R.F.; and Boggs, D.R. (1982) 'Alto: a personal computer'. In Siewiorek, D.; Bell, D.G.; and Newell, A. (Eds.) *Computer Structures: Principles and Examples,* McGraw-Hill Book Company.

Wahlster, W. (1989) User and discourse models for multimodal communication. In Sullivan J.W.; and Tyler S.W. (Eds.) *Architectures for*

Intelligent Interfaces: Elements and Prototypes, Addison-Wesley, Palo Alto, C.A.

Waltz, D.L.; and Goodman, B.A. (1977) Writing a natural language data base system. *Proceedings IJCAI,* pp. 144-150.

Warren, D.H.D.; and Pereira, F.C.N. (1982) An Efficient Easily Adaptable System for Interpreting Natural Language Queries. *American Journal of Computational Linguistics,* Volume 8, no. 3-4, July-December.

Weizenbaum, J. (1966) ELIZA - A Computer Program for the study of Natural Language Communication between Man and Machine. *Communications of the ACM,* Vol 9, No. 1, pp. 36-44.

Wenger, E. (1987) *Artificial Intelligence and Tutoring Systems* Morgan Kaufmann, Los Altos.

Woods, W.A. (1977) Lunar Rocks in Natural English: Explorations in natural language question answering. In Zampoli, A. (Ed.) *Linguistic Structures Processing.* New York: Elsevier North Holland.

Woods, W.A. (1973) Progress in Natural Language Understanding: *Proceedings AFIPS Conference*, 42, AFIPS Press.

Woods, W.A. (1970) Transition Network Grammars for Natural Language Analysis. *Communications of the ACM.* Vol 13. pp.591-606.

Wright, P.; and Monk, A.F. (1989) Evaluation for design. In Sutcliffe, A. and Macaulay, L. (Eds.) *People and Computers V. Proceedings of the Fifth Conference of the British Computer Society on Human-Computer Interaction.* Cambridge University Press. pp. 345-58.

Appendices

Appendix I: Modules of Circuit II

- **Control**

Circuit II, like SOPHIE, uses a control module which receives input to be acted upon. Control has two primary functions: First, all input received through the graphical interface is passed to one of five library modules (MtrLibrary, LngLibrary, PrtLibrary, CxtLibrary and MLibrary) which parse and evaluate the input. Based on this evaluation either the appropriate specialist and/or the circuit simulator is invoked from the appropriate library module. Second, a semantic interpretation of the mouse selections made is returned and displayed in natural language to enhance the meaningfulness of the graphical interaction. Each of the modules and the procedures used by Circuit II are described in more detail below.

- **MtrLibrary**

The meter library contains a number of procedures which are used to control the various functions of the tools which can be applied to the electronic components of the circuit. If one of the tool icons is activated first then one or a combination of the following procedures will be invoked.

PROCMeter

When invoked, this procedure a) instantiates system variables with values and b) calls other procedures in turn to carry out the following tasks:

1. Invokes one of the following procedures:
   a. PROCPlot_V_overlay - highlight the voltage icon.
   b. PROCPlot_I_overlay - highlight the current icon.
   c. PROCPlot_R_overlay - highlight the resistor icon.
   d. PROCPlot_High_overlay - highlight the high resistance fault icon.
   e. PROCPlot_Low_overlay - highlight the low resistance fault icon.
   f. PROCPlot_Open_overlay - highlight the open-circuit fault icon.

g. PROCPlot_Short_overlay - highlight the short-circuit fault icon.

h. PROCPlot_Insert_overlay - highlight the insert random fault icon.

i. PROCStep1 thru Step8 - explanation of steps to take to locate a fault.

depending upon which tool functions has been selected.

2. After one of the procedures described in 1a - 1g has been invoked a check will be made to see if an anaphoric-like mouse action has occurred by invoking the procedure PROCSpecified_Use (in the LngLibrary).

3. When the procedure in 1h has been invoked the normal working context will change to a faulted one by invoking the procedure PROCChangeContextC2ToC3 (in the CxtLibrary). Next a random fault will be inserted into the circuit by invoking the procedure PROCInsertRandomFault (in the FltLibrary) which will modify the MITEYSPICE circuit file and save the changes made by invoking the procedure PROCWriteSPICE (in the FltLibrary).

4. The procedure in 1i can only be invoked in a random faulted context. Its only function is to provide a step by step explanation of the measurements which could have been taken in order to locate the random inserted fault should the users be unable to locate it by themselves.

- **LngLibrary**

The language library contains all the procedures which control the highlighting of icons and mouse selections which perform analogous actions to anaphora, ellipsis and deixis. The order in which these procedures are invoked will depend upon the actions of the user. These procedures perform the following actions:

PROCSpecified_Use

This procedure is invoked when the type of measurement (i.e., voltage, current and resistance representing commands) which is being applied to a

component (an argument) is varied. For example, if a current measurement is being taken through a resistor and the voltage icon is selected, this has the effect of de-highlighting the current icon in favour of the voltage icon before control is returned to the calling procedure.

PROCFault_Use

In a similar manner, this procedure is invoked when the type of fault (i.e., open-circuit, short-circuit, high and low representing commands) which is being applied to a component (an argument) is varied. The currently highlighted icon is de-highlighted in favour of the newly selected one.

PROCEllipsis

Like measurements, a component (an argument) can be varied in favour of another, so for example, if the resistor R1 is currently highlighted and then R3 is selected, then this procedure will be invoked. When this happens, R1 will be de-highlighted in favour of R3. All circuit components can be handled in this way.

PROCFault_Ellipsis

When a particular fault is applied to a component within the hypothetical fault mode, it can be moved on to other components in the same way as previously described for component ellipsis. PROCFault_Ellipsis handles this type of ellipsis when it is invoked.

PROCSpec_Check

As well as handling mouse actions analogous to anaphora and ellipsis, Circuit II also handles deictical selections which are seen as brief conversational digressions in those cases where measurements are being taken. When a component's specification is requested (by depressing the third mouse button over it) the procedure Spec_Check is invoked which returns that component's details. After a short pause the procedure PROCContextSwitch is invoked which decides, depending up the type of

measurement being taken prior to the digression, which canned text procedure (i.e., Volts, current or resistance) to invoke to display the appropriate return message.

- **PntLibrary**

The points library contains the procedures which control the inspection points representing circuit nodes in Circuit II. The inspection points can only be used to take voltage measurements between ground and some other point or between components. Current through nodes cannot be taken since it is a meaningless measurement, and resistance measurements between nodes are not supported. The sequence in which one or more procedures within this library are called will be determined by a user's mouse selections.

PROCPoints

This procedure when invoked instantiates values to variables and then invokes the procedure PROCEllipsis contained in the language library. This checks to see if a previous inspection point has been highlighted previously, and if so de-highlights it in favour of the newly selected one. This procedure also invokes the appropriate procedure within the measurement library which displays a response to the user.

- **ResLibrary**

The resistor library contains all the procedures which control a resistor's activation when measurement requests, i.e., voltage, current or resistance are made.

PROCResistors

When a resistor is activated this procedure checks to see if an elliptical mouse selection has been made on a faulted or unfaulted component. If one has, then either PROCFault_Ellipsis or PROCEllipsis is invoked within the

language library which de-highlights the previously highlighted resistor. The current resistor is then highlighted when control is returned to PROCResistors. If the resistor is faulted then procedures within the fault library are invoked which updates and re-writes the new fault context to the MITEYSPICE file after which the circuit simulator is run to compute new circuit values. Using these new computed values the measurement library is accessed and the appropriate procedure invoked which displays a textual response to the user.

- **TrnLibrary**

The transistor library handles all mouse selections made over circuit transistors for voltage, current and resistance measurements that can be applied to them. The transistors are different from other circuit components in that they have more than one part which can be activated at a time, depending upon the type of measurement being taken. For example, a voltage measurement across a transistor can be taken in two different ways. The inspection points to which its terminals are connected can be highlighted or its body (for example, the base-emitter) can be selected instead. Current and resistance measurements can only be applied to the body of the transistor and to one part (for example, the base, collector or emitter) at a time.

PROCTransistors

This procedure checks to see if a transistor has been selected and if an elliptical mouse selection has been made over its part(s). The type of elliptical mouse selection made (normal or faulted) is also checked and the appropriate procedure is invoked from within the language library. If the component is unfaulted then the previously highlighted component is de-highlighted in favour of the newly selected one before control is returned to the calling procedure. However, if the component is faulted then when

control is returned to PROCTransistors the procedure corresponding to the way that the component is faulted is invoked from within the fault library which updates and writes the MITEYSPICE file. The circuit simulator is run to re-compute the circuit values before control is returned to PROCTransistors. Depending upon the type of measurement currently being taken, PROCTransistors then calls the appropriate procedure to display the canned text response to the user.

- **CapLibrary**

The capacitor library contains procedures which control the behaviour of capacitors used by Circuit II. The current implementation of Circuit II only allows voltage measurements to be taken across capacitors and does not support current or resistance measurements. Also, at present, they cannot be faulted unlike other circuit components.

PROCCapacitors

This procedure checks to see if a capacitor has been activated and then checks to see if an elliptical mouse selection is being made. If an elliptical mouse selection is being made then PROCEllipsis, within the language library, is invoked which de-highlights the previously highlighted component before returning control to PROCCapacitors. PROCCapacitors then highlights the newly selected capacitor and then calls the procedure PROCC_Volts within the measurement library which displays the appropriate textual response to the user.

- **CxtLibrary**

The context library handles all three contexts supported by Circuit II. The first context supported and displayed to users is a screen display of a normal working circuit. From this context users may digress either to the second context, a hypothetical fault mode, or to a third context, a randomly faulted

mode. The procedures within the context library save all system variables and screen displays when a digression to another context is signalled and restores the old context and variables when it is resumed again.

PROCContextCheck

This procedure, when invoked, checks the current context status and changes it when signalled to by calling one of the following procedures a) PROCChangeContextC1ToC2 b) PROCChangeContextC2ToC1, c) PROCChangeContextC1ToC3 and d) PROCChangeContextC3ToC1.

PROCChangeContextC1ToC2

When this procedure is invoked, moving the user to a hypothetical fault context, the system variables of the normal working circuit context are saved and the previously highlighted icons are all reset. There are two ways of signalling to the system to digress to this context, either by selecting a fault icon or a component with the second mouse button. Within this new context users may pose hypothetical "What...If..." types of graphical questions to Circuit II and have them answered. When a user is ready to return to the previously interrupted context they do so by pressing the first mouse button over the return label to invoke the procedure PROCChangeContextC2ToC1.

PROCChangeContextC2ToC1

This procedure when invoked resets the previously selected icons and restores all system variables and highlighted icons within this context. In addition, an appropriate return message to assist the user's memory in resuming the interrupted context is also displayed.

PROCChangeContextC1ToC3

When a user wishes to move to a randomly faulted context to try to locate an unknown fault, they do so by selecting the random inserted fault icon with the second mouse button. This topic shift is not seen as a brief digression to a related context, rather it is viewed as new context which is totally

unrelated to the normal working circuit. So, when the user signals a move to this context no system variables are saved. When a user decides that they want to return to a normal working circuit they do so by clicking on the first mouse button over the return label marked "Norm Circuit".

PROCChangeContextC3ToC1

This procedure, when invoked, returns the user to a normal working context by resetting all system variables and de-highlighting all previously activated icons.

PROCChallengeAssertion

This procedure handles all 'challenges' made by a user regarding the correctness of a response given by the system. To challenge the system's response they can click on either a '√' or a 'x' and then click directly over the textual response given by the system. These graphical actions are analogous to the natural language expressions "Is that right?", "Is that wrong?" or "What should it be?" depending upon whether a '√' or 'x' was selected. In response to these graphical actions a 'challenge' window is displayed within which the icons selected and their natural language equivalent is displayed. This procedure invokes the next procedure PROCSupportMeasurement.

PROCSupportMeasurement

This procedure displays another window within which the system responds to a user's challenge by informing them that the measurement given is either correct or incorrect with respect to a working circuit. PROCSupportMeasurement also identifies the component whose measurement is being 'challenged' and calls the appropriate procedure from within the measurement library.

- **FltLibrary**

The fault library contains all the procedures which can be invoked to fault circuit components in a range of ways. The types of faults currently supported by Circuit II include open and short-circuiting resistors, capacitors, setting resistors to a high or low value and inserting an unknown fault. This library also includes procedures which when invoked modify and write MITEYSPICE files as well as running the circuit simulator.

PROCHighFault

This procedure when invoked sets the value of a chosen resistor to a high value. It does this by using two string variables, one which identifies the unfaulted component as described within the MITEYSPICE file and one with a high value which will be used to replace it. A third string variable is assigned textual information about the nature of the fault, for example, "load resistor OR1 goes high" which will be used by the measurement library when invoked. The procedure PROCWriteSPICE (in the FltLibrary) is then invoked to up-date the MITEYSPICE file before running the circuit simulator to compute new values for the circuit nodes.

PROCLowFault

When invoked this procedure performs the same operations as PROCHighFault except that modification made to the MITEYSPICE file sets the resistor to a low value.

PROCOpenFault

When invoked this procedure performs the same operations as PROCLowFault except that modification made to the MITEYSPICE file sets a resistor or the parts of a transistor to an open-circuit value.

PROCShortFault

When invoked this procedure performs the same operations as PROCOpenFault except that modification made to the MITEYSPICE file sets a resistor or the parts of a transistor to a short-circuit value.

PROCInsertRandomFault

Invoking this procedure performs the same operation as the other fault procedures already mentioned except that the nature of the modification made to the MITEYSPICE file is unknown to the user.

PROCWriteSPICE

This procedure when invoked first reads into memory each string of data within the MITEYSPICE file which uniquely identifies individual circuit components. As each string is read it is compared against the string of the component to be modified. When a match is found the unfaulted component string is replaced by the modified one which represents the fault to be inserted. A copy of the MITEYSPICE file is then written to main memory and the circuit simulator, MITEYSPICE, is invoked to compute new circuit values using the modified file as input. Computed values returned by MITEYSPICE are then re-read into system variables and used by other procedures to present the data to the user.

- **MLibrary**

The measurement library contains all procedures which display canned text answers and prompts in response to the graphical questions posed by users within the three contexts supported by Circuit II. Each procedure handles a range of textual responses for a particular class of circuit component (capacitor, transistor and resistor) and type of measurement (voltage, current and resistance) being taken. Since each procedure performs a similar function only one, PROCC_Volts, will be described in detail below.

PROCC_Volts

This procedure when invoked first determines if the status of the current context is faulted or unfaulted. If the context is unfaulted then an array holding normal working inspection point data is used to identify which circuit nodes have been activated. Three different grades of textual responses are given to the user, that is, either full, abbreviated or numerical answers. The grade of response given is determined by a numeric variable. When the procedure is first invoked the variable has the value '0' which triggers a full textual response (for example, "The voltage across...") to be displayed to the user before being incremented by '1'. However, when another measurement is taken and the procedure is invoked again an abbreviated form of the response (for example, "Across the...") is displayed instead and the variable is incremented from '1' to '2'. When the procedure is invoked for a third time only the numerical value is given (for example, "2.45 volts") and the variable is incremented again from '2' to '3'. Unless the type of measurement being applied changes, say from voltage to current, then each time the procedure is invoked the textual response will vary between an abbreviated and numerical response.

In addition, the measurement library contains other procedures which are invoked from within the context library when a user challenges the correctness of an answer given by the system. Different procedures exist to support the system's response to voltage, current and resistance measurements given. Since these groups of procedures perform a similar function only one, PROCSupportR_Volts, will be described in detail below.

PROCSupportR_Volts

This procedure deals with circuit resistors and when invoked gives the normal working voltage across the selected resistor within the support window displayed. The previously given measurement across the selected resistor and its normal working voltage are compared and if they are the

same the system informs the user that the measurement is correct. Where a descrepancy exists between the two measurements, then the user is informed that they are incorrect.

- **RpLibrary**

The procedures within the replace library can only be accessed when a user is troubleshooting the electronic circuit within a randomly faulted context. After taking a series of measurements to try to locate the nature of the fault, a user may ask the system to replace a component which they think is faulty or suspect. They do this by selecting the 'soldering iron' icon with the first mouse button which a) changes the shape of the mouse pointer to the word "CHANGE" and b) invokes the procedure PROCReplace. Before allowing the component to be replaced the system will ask the user to describe the manner in which the component is faulted. If they do not correctly describe the fault or if that particular component is not faulted at all then the system will not replace the component. However, if the nature of the fault is correctly described to the system then the component will be replaced and, where appropriate, the user will be informed that they have located the randomly inserted fault.

PROCReplace

This procedure, when invoked, allows the user to select the suspected faulty component by placing the mouse pointer, displaying the word "CHANGE", over it. When this has been done the procedure PROCDisplayComponent is invoked.

PROCDisplayComponent

This procedure displays the selected component next to the soldering iron, identifies it with a label and then asks the user to describe in what manner the component is faulted by invoking the procedure PROCSelectFault.

PROCSelectFault

This procedure allows the user to describe the manner in which the chosen component is faulted by selecting a fault icon (i.e., open, short, high or low) and applying it to its parts.

PROCAcceptReject

To decide whether or not the user has correctly identified the faulted component this procedure is invoked. PROCAcceptReject does this by comparing the known fault against the suggested one and if they match the component is replaced and a textual response to this effect is displayed. If they do not match then the user is informed and allowed to continue taking more measurements.

- **HypLibrary**

The procedures within the hypothesis library are only invoked when a user is troubleshooting within a random faulted context. This library contains procedures which a) maintain a history list of all measurements taken within this context, b) check the validity of posed hypotheses and c) present a critique of the user posed hypothesis when it conflicts with observed measurements.

PROCIncrHistLst

Each time this procedure is invoked six parameters are passed to it which contain information about the measurement just taken. These parameters contain the following information: 1) the type of measurement taken (i.e., voltage, current or resistance), 2) the component's identification number if it is a capacitor, resistor or if a single part of a transistor is being referred to, 3) the component's identification number if more than one part of a transistor is being referred to, 4) the observed measurement, 5) the array parameters from which the hypothesized measurement is calculated and 6) the normal working value of that component.

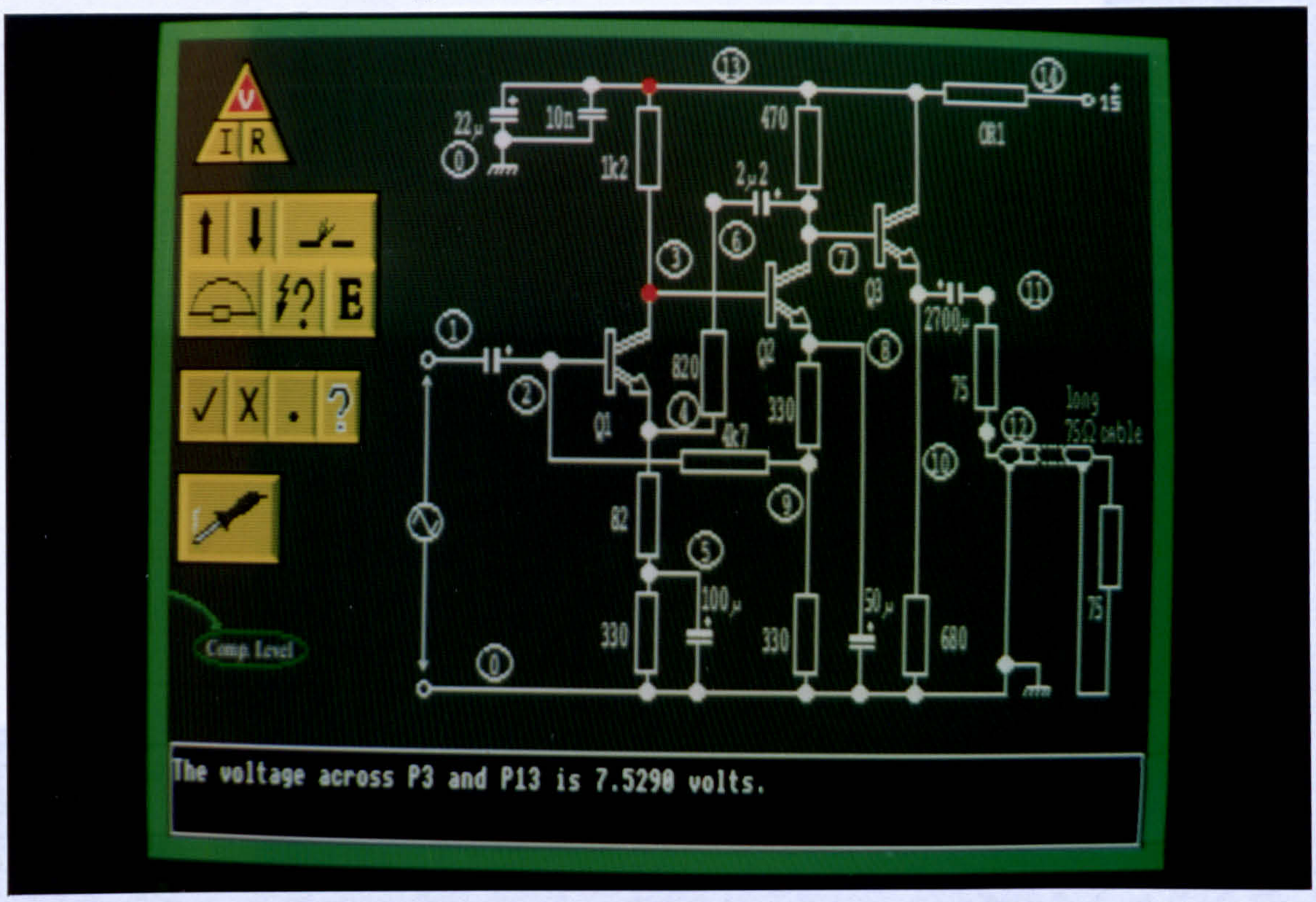
PROCHypothesisCheck

This procedure checks the validity of a posed hypothesis by comparing each measurement observed against what it would be under the hypothesized fault. If all of the measurements compared are the same then the system informs the user that their hypothesis is consistent with the measurements observed so far.

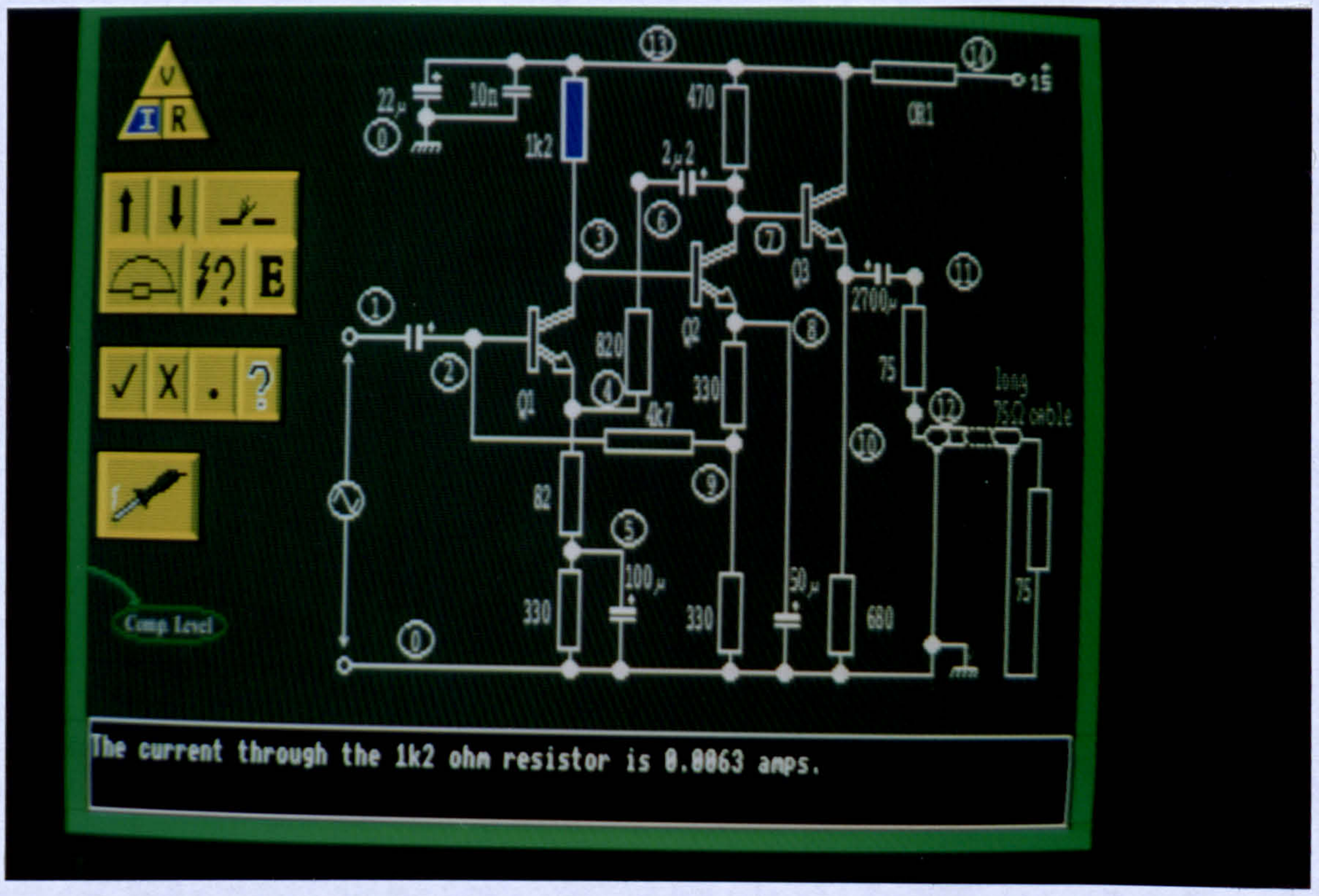
However, if they are different, then the user's hypothesis as to what the fault might be is incorrect. Every measurement which the system finds that is in conflict with the user's hypothesis is brought to their attention. The system does this by presenting the user with three pieces of information: a) the observed measurement, b) the value of that measurement under the proposed hypothesis and c) what that measurement would be in a normal working circuit. After the information has been displayed, the user may continue to review other measurements which conflict with their hypothesis or they may abandon the system's critique and resume taking more measurements.

Appendix II: Examples of Circuit II screendumps

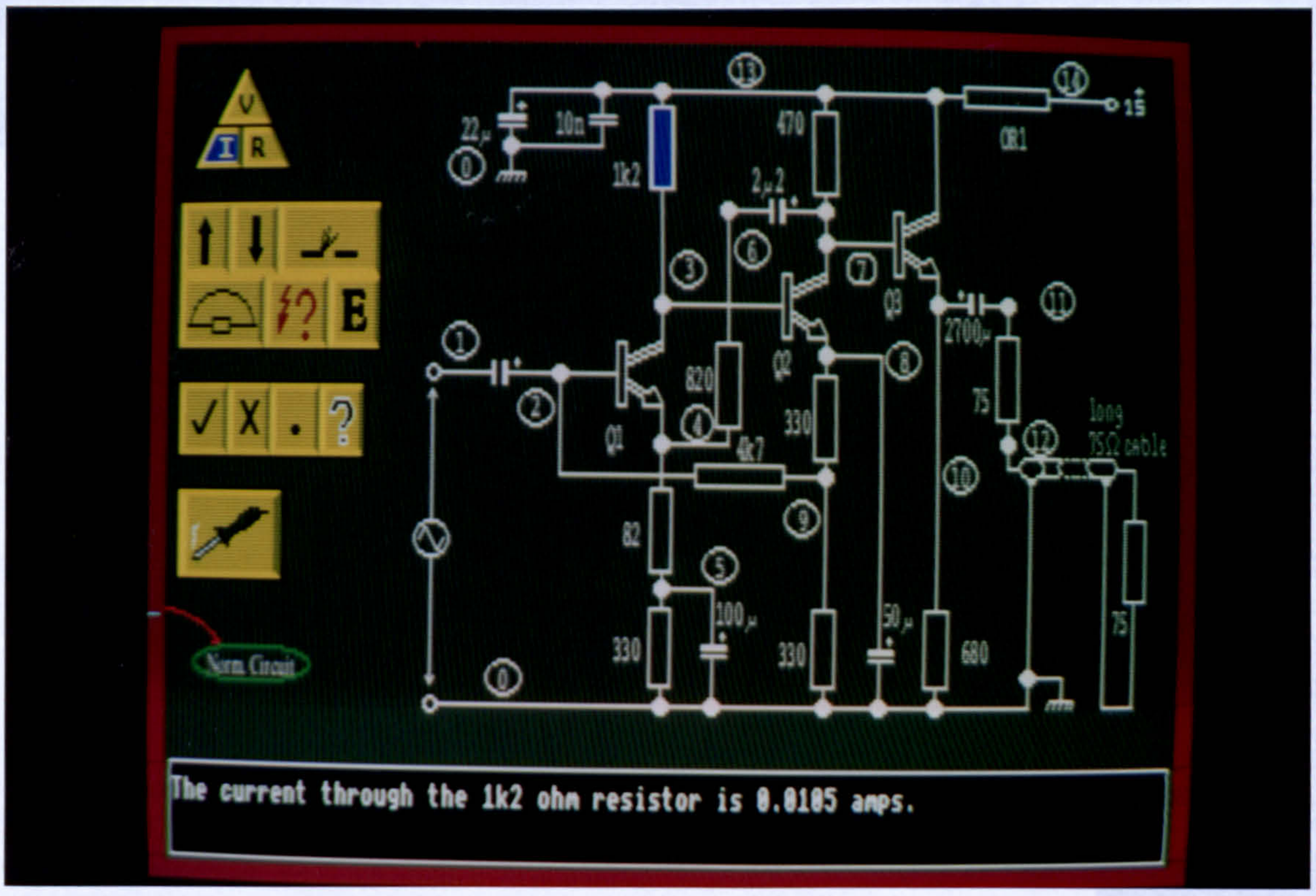
Measuring voltage in a normal working circuit



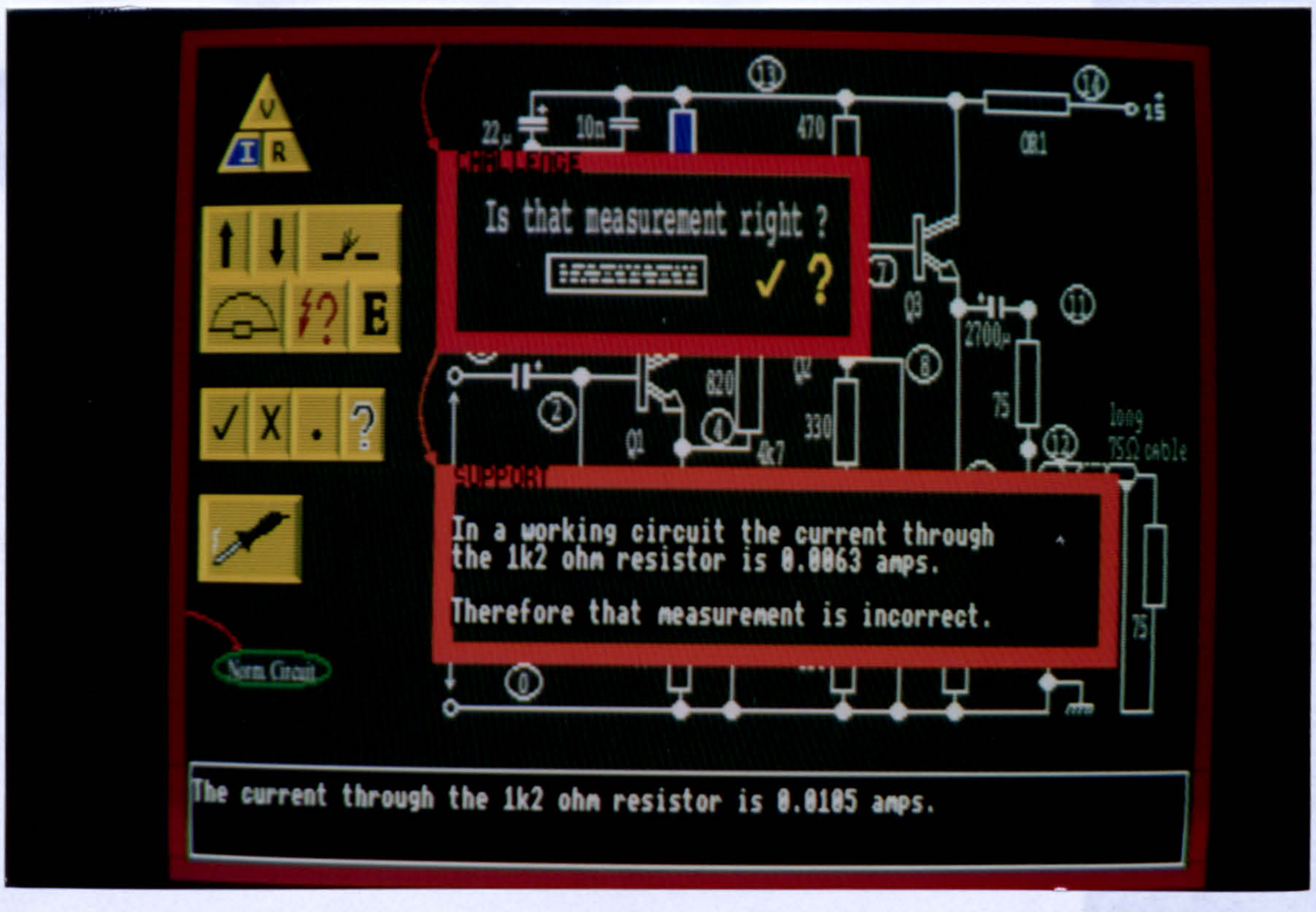
Measuring current in a normal working circuit



Measuring current in a faulted circuit

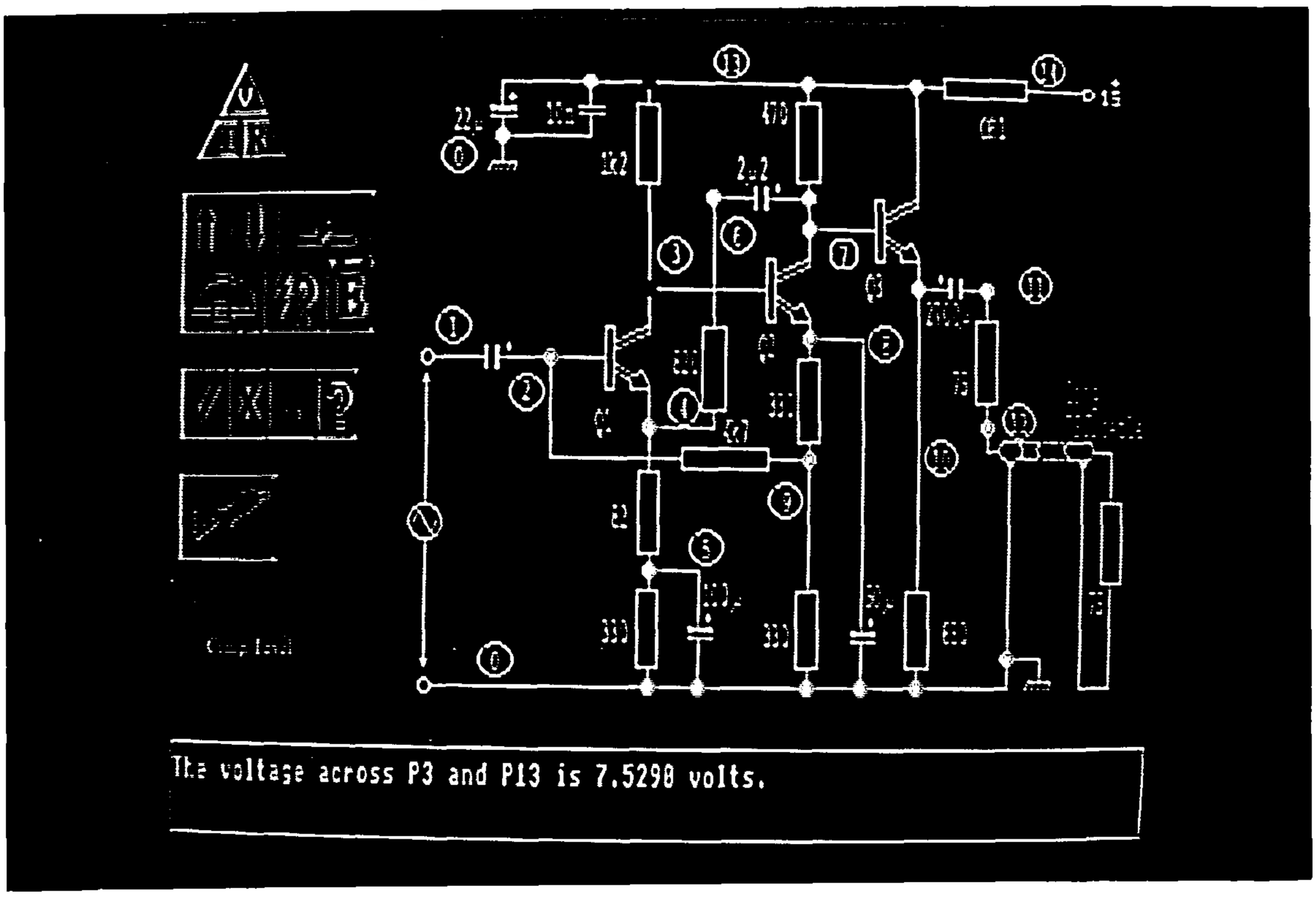


Checking what the correct current measurement should be.

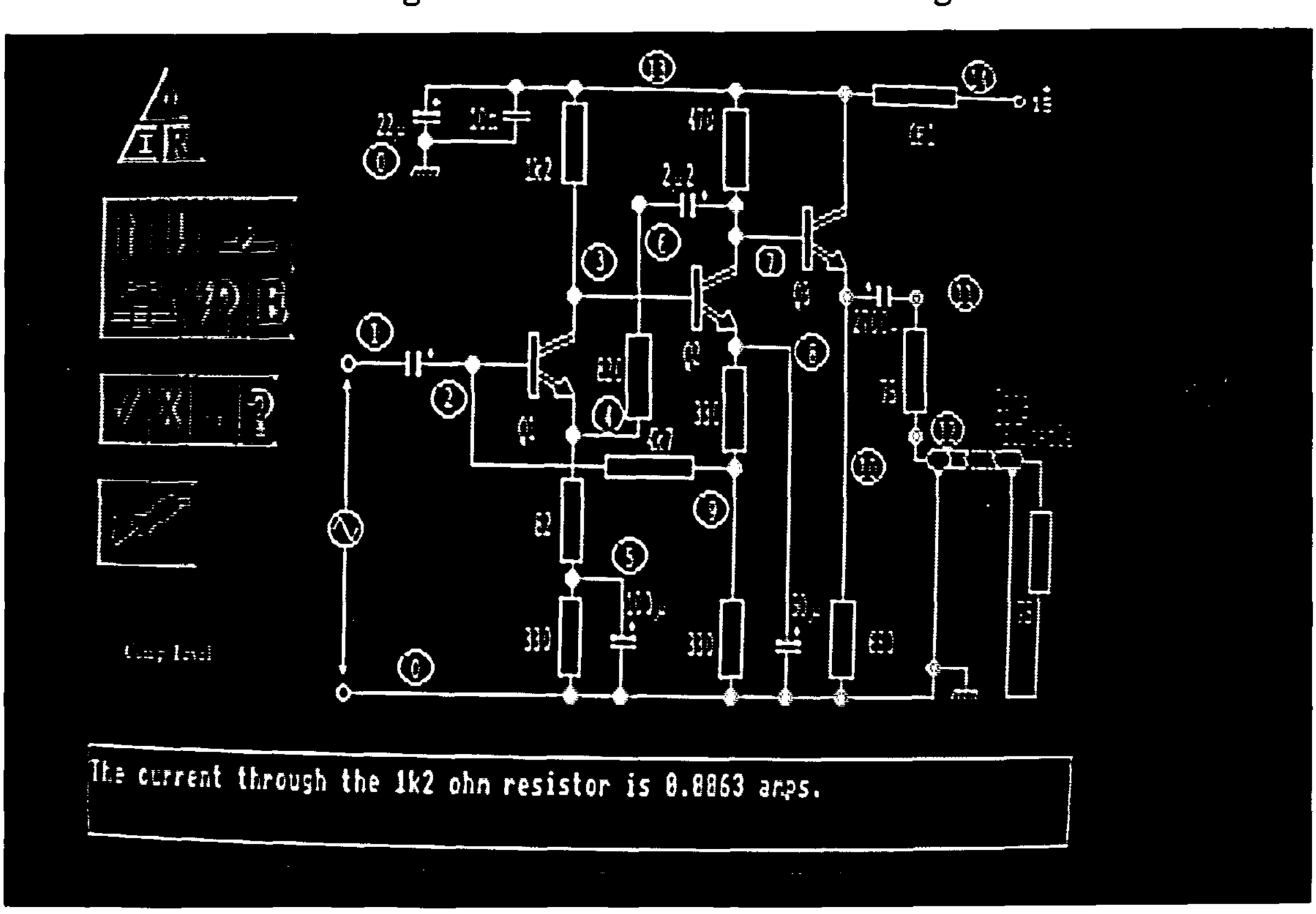


Appendix II: Examples of Circuit II screendumps

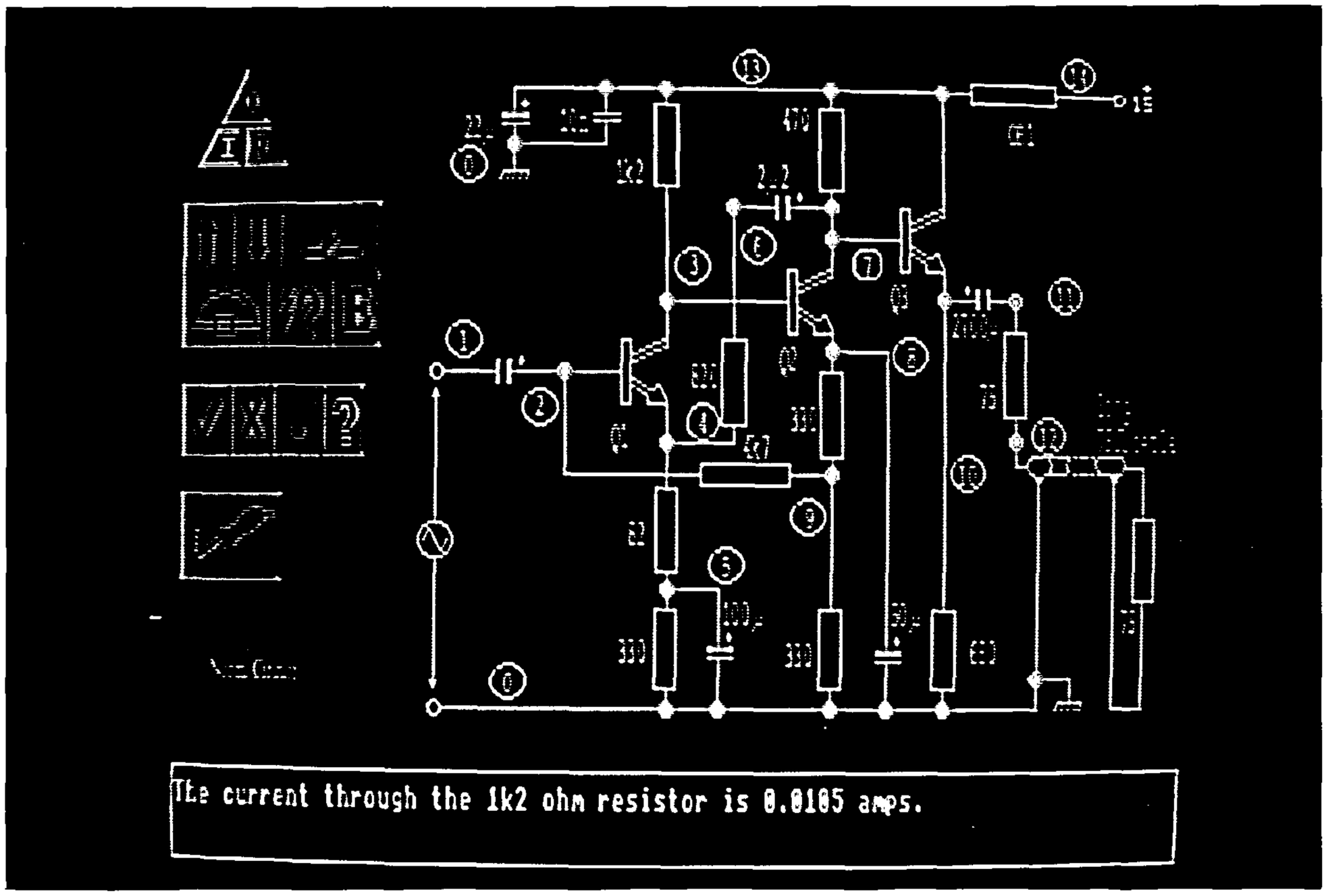
Measuring voltage in a normal working circuit



Measuring current in a normal working circuit



Measuring current in a faulted circuit



Checking what the correct current measurement should be.